Master's Thesis

Master's Degree in Automatic Control and Robotics

Robust 3D IMU-LIDAR Calibration and Multi Sensor Probabilistic State Estimation

Report

22nd of June of 2020

Author: Alejandro Mora Martínez

Director: Juan Andrade-Cetto

Secretary: Alberto Sanfeliu Cortés

Call: 2019-2020 Spring Semester

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Scaled Robotics

Abstract

Autonomous robots are highly complex systems. In order to operate in dynamic environments, adaptability in their decision-making algorithms is a must. Thus, the internal and external information that robots obtain from sensors is critical to re-evaluate their decisions in real time. Accuracy is key in this endeavor, both from the hardware side and the modeling point of view.

In order to guarantee the highest performance, sensors need to be correctly calibrated. To this end, some parameters are tuned so that the particular realization of a sensor best matches a generalized mathematical model. This step grows in complexity with the integration of multiple sensors, which is generally a requirement in order to cope with the dynamic nature of real world applications.

This project aims to deal with the calibration of an inertial measurement unit, or *IMU*, and a Light Detection and Ranging device, or *LiDAR*. An offline batch optimization procedure is proposed to optimally estimate the intrinsic and extrinsic parameters of the model. Then, an online state estimation module that makes use of the aforementioned parameters and the fusion of LiDAR-inertial data for local navigation is proposed. Additionally, it incorporates real time corrections to account for the time-varying nature of the model, essential to deal with exposure to continued operation and wear and tear.

Keywords: sensor fusion, multi-sensor calibration, factor graphs, batch optimization, Gaussian Processes, state estimation, LiDAR-inertial odometry, Error State Kalman Filter, Normal Distributions Transform.

Contents

List of Figures

List of Tables

Notation

- Functions are denoted by small case letters with parenthesis (e.g. $f()$)
- Scalars are represented by small case, non-bold greek letter (e.g. λ) or italic letters (e.g. a)
- Vectors are denoted by bold, small case letters (e.g. $\mathbf{v}$), and matrices by bold, capital letters (e.g. $\mathbf{R}$). Vectors may also be denoted by an arrow on top of a small case letter (e.g. $\vec{p}$)
- Numeric subscripts in matrices indicate their dimension. A single number denotes square matrices (e.g. $\mathbf{I}_3$ means that $\mathbf{I}$ is a square matrix of size 3), while two numbers denote non-square matrices (e.g. $\mathbf{I}_{6x3}$ means that $\mathbf{I}$ is a matrix with 6 rows and 3 columns)
- A hat over a variable denotes an estimation of said variable (e.g. $\hat{x}$ is an estimate of x). A bar over a variable denotes its mean (e.g. $\bar{x}$ indicates the mean value of x)
- Sets are represented by scripted capital letters (e.g. $\mathcal{M}$)
- Capital letters in subscripts and superscripts denote a frame of reference (e.g. g_W indicates that variable g is referred to frame W)
- Time is always represented by t. The timestamp associated to a discrete instant k is indicated via superscript or subscript (e.g. t_k). t may also be used for 3D translations $\mathbf{t} \in \mathbb{R}^3$, in which case it will be explicitly mentioned.

CHAPTER 1

Introduction

Bestowing a robotic platform with the ability to parse sensory information and reason about its internal state and external environment is a essential step towards autonomous behavior. Education [1], medical and social assistance [2; 3] and safety and rescue operations [4] are only some examples of applications to benefit from their capabilities.

Market studies foretell robotic technologies to disrupt traditional solutions across the aforementioned fields during the upcoming decades [5], despite the challenges they have to face. The complexity of implementing robotic systems stems from multiple points of view. From the engineering perspective, one of the hardest challenges is the integration of all the layers that comprise this decision-based paradigm; in essence, the behavior of robotic manipulators and mobile platforms is that of a control loop, in a comparable fashion to humans.

A perception layer gathers multi-modal data from sensors and processes it to extract information. Part of it can be used to build a map of the surroundings and self-localize in that environment [6]. At that point it is possible to plan a path to achieve a certain goal [7] and apply motion control [8] to reach it, thus closing the loop. Robust calibration between sensors enable such capabilities.

The amount of literature that explores how to deal with each of these topics, which are still open, is enormous. For the sake of clarity, from this point onward the focus on this dissertation will be put on the foundation of autonomous robotic systems: *perception* and *localization*, or sensorial data processing and its use in determining the robot's pose within the environment. Providing an autonomous system with such capabilities is crucial to develop higher-level layers.

1.1 Motivation and objectives

There exists a wide assortment of systems and processes in which quality control and inspection tasks are of interest, whether it be for delivering information to the user or to apply a feedback action [9; 10]. Accuracy, reliability and repeatability are some of the must-have properties of measurement tools that are to be employed for these tasks.

Within this context, the devices that allow a robotic platform to measure its internal state and its surroundings are sensors. Most commonly, the distribution of such systems constitute either human-operated or autonomous but static solutions.

However, there are situations in which an unmanned, mobile robot carrying the necessary measurement tools can provide additional value, such as operation in hazardous or inaccessible environments [11], remote pipeline inspection [12] or ship hull maintenance [13]. Through the

combination of proprioceptive and exteroceptive sensors the robot can not only navigate the environment [14], but also collect information of interest. RGB cameras, ultrasound or time of flight devices are some examples of the former, while inertial measurement units and torque sensors are some of the most common of the latter.

The broad range of available sensing devices and the fact that they can complement each other has motivated extensive literature on sensor fusion, which deal with multi-sensor configurations. These devices are studied as a mathematical model; however, it is challenging to analytically deal with time-varying conditions and characterize non-linear sources of noise [15].

In the end, sensor models hold a certain degree of uncertainty and parameter-dependency, and the way in which these are adjusted is what constitutes the calibration process. The accuracy and long-term reliability of an autonomous mobile platform are directly tied to the quality of the intrinsic calibration of each individual sensor, as well as to the extrinsic calibration of the kinematic relationships between them [16].

Note that these requirements become twofold when the robot does not only use sensor data to navigate the environment. Furthermore, due to the nature of wear and tear degradation caused by transportation, mechanical vibrations and unexpected impacts (amongst others), the calibration procedure should be as reproducible as possible, and could ideally be performed online during robot operation.

This dissertation intends to delve further into multi-sensor calibration and probabilistic state estimation on the basis of sensor fusion theory. A proprioceptive sensor, an Inertial Measurement Unit (IMU), and an exteroceptive sensor, a Light Detection and Ranging (LiDAR), are considered within the scope of this work. The author sets to achieve the following objectives:

1. Calibration of the intrinsic parameters of the IMU model and the extrinsic parameters of the transformation between the IMU-LiDAR sensor pair.

2. Determination of a calibration procedure that minimizes the amount of assumptions required to be applicable, in order to increase its versatility in a generalized context.

3. Development of a probabilistic state estimation framework that includes the optimized calibration parameters to perform online localization.

4. Extension of the state estimation algorithm to consider updating calibration parameters (both intrinsic and extrinsic) online.

1.2 Contents

In section 2, the hardware considered in this work and the problems addressed, derived from sensor fusion, will be described in higher detail. In section 3 the implementation of an off-line, batch optimization algorithm for IMU-LiDAR calibration will be formulated, along with all the theoretical background in which is based.

In section 4, the output of section 3 will be used to formulate an online state estimator than can fuse inertial and LiDAR-based data to perform odometry, and will be extended to consider online updates of said parameters. In parallel to the structure of section 3, the mathematical background and the code implementation will also be described.

Section 5 will present the experimental results of the calibration and state estimation algorithms using publicly available datasets and own-collected data, while benchmarking against other comparable state-of-the-art techniques. Sections 6 and 7 will reflect on the social, economical and environmental impacts of the project. Finally, section 8 gathers the conclusions on the developed work and discusses the guidelines for future projects.

CHAPTER 2

Preliminaries

In general, multi-sensor configurations share common traits and challenges, regarding issues as synchronicity and data association, amongst others. However, the way in which they are studied intrinsically depends on the selection of hardware, according to their mode of operation and the nature of the data they are able to harvest.

In this section, a description of the sensors that will be considered in this work will be provided, as well as a formal introduction to sensor fusion and the specific problems that are aimed to be solved.

2.1 Inertial Measurement Unit (IMU)

One of the most extensively used proprioceptive sensors in mobile robotics are the *inertial measurement units* or IMU. Low-cost, non-invasiveness, compact sizes and high data rates are some of the factors that endorse their popularity and enable real-time applications [17].

This device consists of a tri-axial accelerometer and a tri-axial gyroscope that allow measuring the linear acceleration and angular velocity of the sensor with respect to an inertial frame, respectively. One can assume the Earth to be an inertial frame by neglecting its rotation [18], which is reasonable in most applications.

There exist 9 degrees-of-freedom (DoF) IMU models that incorporate a magnetometer for achieving drift-free 3D orientation tracking [19], but because of their difficult calibration and sensitivity to time-varying disturbances, they are usually disregarded in favor of the aforementioned 6 DoF model.

Consequently, the vastly accepted measurement model of an IMU is the following [18]:

$$\begin{aligned} \hat{\mathbf{a}}_I &= {}^I\mathbf{R}_{IW}{}^T(\mathbf{a}_W - \mathbf{g}_W) + \mathbf{b}^a + \eta^a \\ \hat{\omega}_I &= \omega_I + \mathbf{b}^\omega + \eta^\omega \end{aligned} \tag{1}$$

where $\hat{\mathbf{a}}(t)$ and $\hat{\omega}(t)$ are the linear acceleration and angular velocity measured by the sensor, respectively. The real values, $\mathbf{a}(t)$ and $\omega(t)$, are affected by slowly-varying bias terms $\left(\mathbf{b}^a(t), \mathbf{b}^\omega(t)\right)$ and additive, zero-mean Gaussian white noise $\left(\eta^a(t), \eta^\omega(t)\right)$.

Superscripts $\bullet^a$ and $\bullet^\omega$ denote variables associated to the accelerometer and gyroscope, respectively, while the subscripts $\bullet_I$ and $\bullet_W$ indicate that the values are referred to the local (IMU) or global (world) frames, respectively as well. Finally, ${}^I\mathbf{R}_{IW}{}^T(t) \in \mathcal{SO}(3)$ represents

the transposed rotation matrix defined in the IMU frame that transforms the local orientation to the global frame. Note that all variables in (1), with the exception of gravity ($\mathbf{g}_W$), are time-dependent, but its notation has been dropped for clarity's sake.

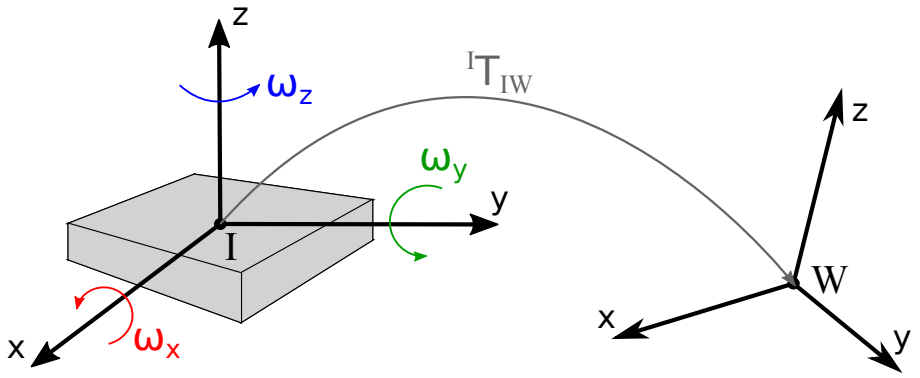

Figure 1: IMU local frame and transformation matrix ${}^{I}\mathbf{T}_{\mathbf{IW}} \in \mathcal{SE}(3)$, composed by a rotation ${}^{I}\mathbf{R}_{\mathbf{IW}}{}^{T} \in \mathcal{SO}(3)$ and a translation $\mathbf{t}_{\mathbf{IW}} \in \mathbb{R}^3$, from the local frame I to the global frame W.

Accelerometer and gyroscope biases $\mathbf{b}^{\mathbf{a}}, \mathbf{b}^{\omega} \in \mathbb{R}^3$ are Gaussian variables that reflect an offset error in the IMU measurements. In other words, the non-zero output of the sensor even in the absence of rotation or acceleration is modeled by these parameters. What's more, their value is affected by long term drift, represented as a Brownian motion [18].

2.2 Light Detection and Ranging (LiDAR)

The necessity of building geometric representations of the environment in applications such as farming crops classification [20], airborne mapping [21; 22], virtual reality or 3D structures modeling [23], and particularly autonomous driving [24], has motivated the introduction of time of flight (ToF) sensors in robotic platforms. By measuring the elapsed time between the emission of a pulsed laser signal at a known frequency and its reception through a photoelectric detector using a high-frequency clock pulse, these devices are able to determine the distance separating the emitter and the surface that deflected the laser [25].

Even though, as discussed in [25], there exist different LiDAR models according to their working principle, the most extended ones are ToF-based devices whose laser direction can be shifted by a MEMS-based vibrating micromirror [26]. The scan of a single ToF sensor (or an array of ToF's) will result in a two-dimensional (or three-dimensional) sparse representation of its surroundings, in the form of a point cloud (Figure 2).

Each point p_i is described by its 3D coordinates $\{x_i, y_i, z_i\}$ and the associated laser's reflection intensity I_i. The significant amount of information embedded in this representation of the environment has attracted a lot of attention; there are still open lines of investigation on how to use this data to perform segmentation [27], classification [28; 29], tracking [30] and point cloud matching [31], driven by their potential applications.

The high-frequency nature of the laser sweeps results in large volumes of data, albeit real-time operation can be achieved by exploiting the sparse nature of the point clouds [32]. This information is also robust against changing lighting conditions, which makes its performance agnostic

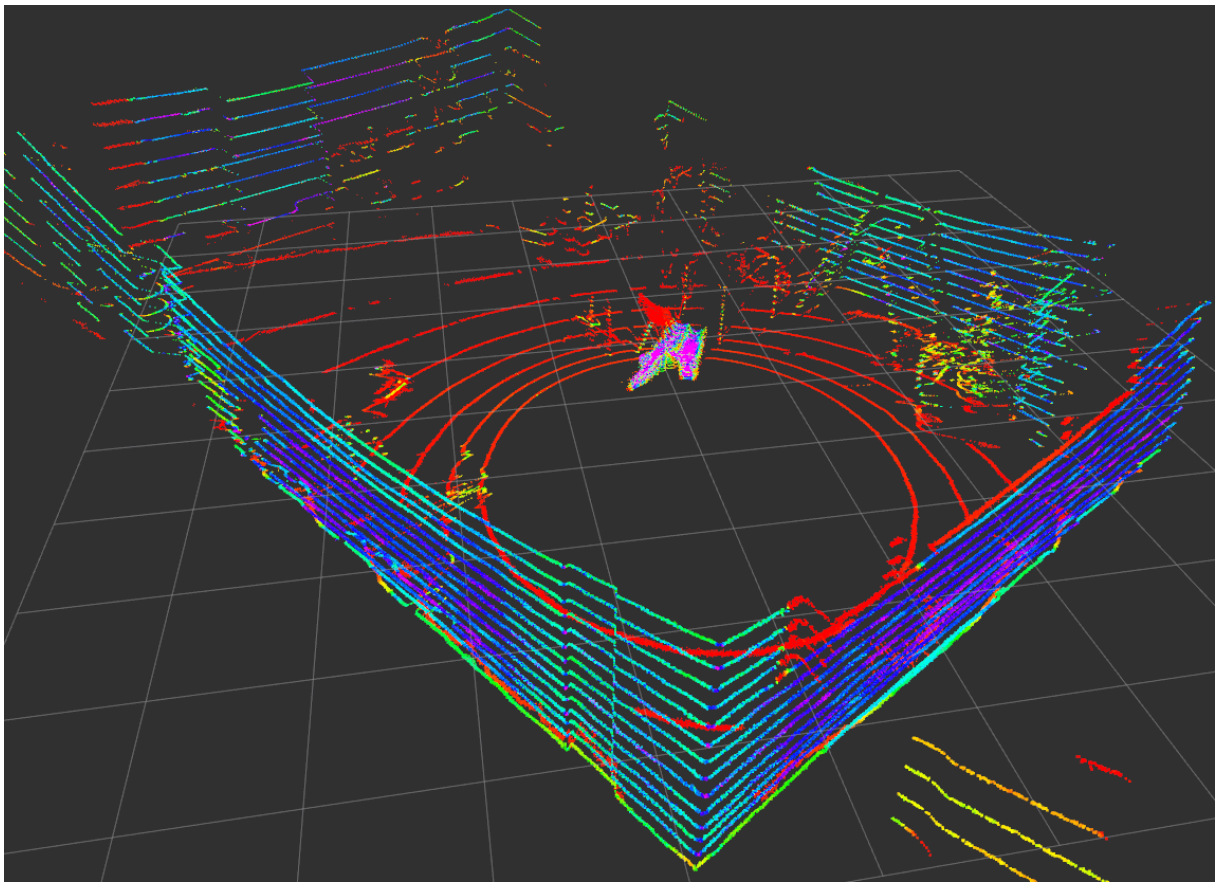

Figure 2: LiDAR 3D point cloud representation of an indoor room. The color of each point denotes the intensity of the laser's reflection.

to indoor or outdoor applications, and a very interesting option to complement camera-based data [33].

It is worth noting that, due to their principle of operation, ToF-based LiDARs show limited capabilities to detect specular surfaces, as for example glass. Since the reflection of an incident laser waveform happens with little dispersion, it is very hard for the photoelectric detector of a LiDAR to capture the reflected beam unless it is oriented almost perpendicularly to the surface in the first place.

2.3 Sensor fusion

The concept of *sensor fusion* denotes the process of merging data from a set of sensors $S = \{S_1, S_2, ..., S_N\}$ such that the uncertainty of the process at hand (navigation, path planning, task performance, ...) may be reduced [34], or to derive information that otherwise may not be perceived. The association of these sensors can be classified according to how their data is combined and which is their communication scheme (Figure 3).

2.3.1 Classification according to data merging strategy

Competitive or *redundant* sensors are those which provide the same kind of information about the world, whether they are multiple sensors or measurements from the same one at different

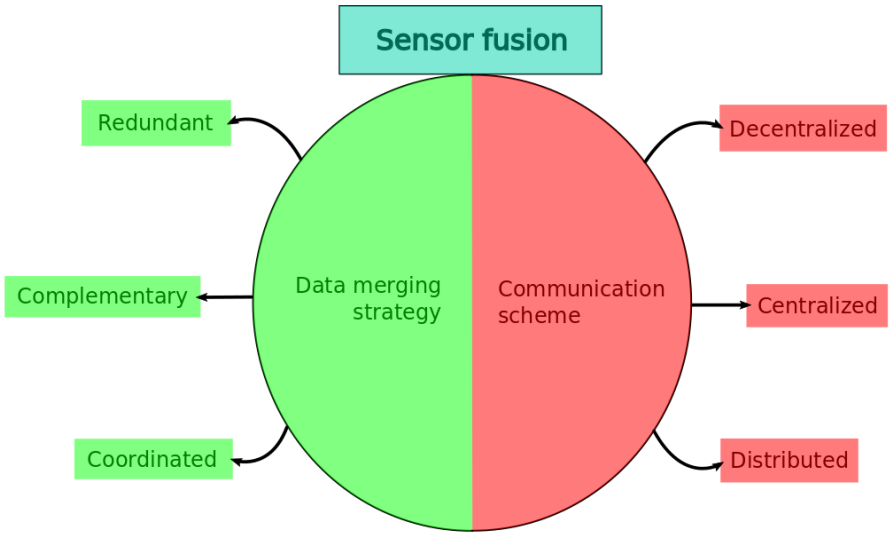

Figure 3: Classification of sensor fusion approaches according to their data combination strategy and communication scheme.

time instants. Redundant configurations can work with heterogeneous data sources and confer robustness to the system [35].

On the other hand, a sensor configuration is *complementary* if the sensors provide *disjoint* information about the environment; this resolves the incompleteness of data in situations where the scene is not fully observable by each individual sensor.

Finally, a *coordinated* or *cooperative* sensor network is that which uses data from each independent sensor to derive higher-level information that would not be available from single sensors alone. In contrast to a redundant scheme, cooperative fusion tends to cause loss of accuracy and reliability due to inaccuracies that may affect each individual sensor [35].

The sensor fusion scheme designed in this dissertation constitutes a **redundant** network with underlying **complementary** properties. While the objective pursued with this configuration is to minimize the uncertainty of the measurements for localization purposes, the mixed nature of proprioceptive (IMU) and exteroceptive (LiDAR) sensors will enable dealing with instances in which part of the data is unavailable or corrupted [36].

2.3.2 Classification according to communication scheme

In a *decentralized* multi-sensor configuration there exists no communication between the sensors. Thus, each device's data is processed independently from the rest of sensors in the network. In contrast, in a *distributed* communication scheme the sensors exchange information at a certain rate.

A particular case of the distributed scheme is a **centralized** network, in which the inter-sensor communication takes place through a central node [34]. This is the architecture implemented in this dissertation's robotic platform: an on-board computer is responsible for receiving data from the sensors in the network (in this particular case, the IMU and the LiDAR) and processing it, enabling data fusion techniques.

Distributed systems show a particular trait, which becomes of pivotal importance for high-frequency applications: **data synchronization**. It is imperative to maintain temporal coherence in order to process data from multiple sources [37]. Since the communication established between the on-board computer and the sensors does not include a synchronization module by default, this issue will have to be explicitly addressed in the software implementation, which will be detailed in sections 3 and 4 (more specifically, in subsections 3.6 and 4.4).

2.4 Intrinsic and extrinsic calibration

Modeling a sensor consists in defining its input-output behavior through mathematical expressions, without loss of generality. This step usually introduces parameters that characterize the relationship between the sensed quantity and the environment: these are referred to as **intrinsic** parameters.

When working with a multi-sensor system, however, modeling includes the characterization of the interactions between sensors as well. The expressions that describe the inter-sensor kinematic relationships introduce **extrinsic** parameters.

Thus, the calibration process consists in tuning the values of the aforementioned intrinsic and extrinsic parameters of a model. This is achieved by minimizing the divergence between the expected behavior of the system, given by the assumed model expressions, and the observed behavior, derived from experimental data collected using a specific realization of said model.

The scope of this work contemplates two sets of intrinsic parameters, as briefly announced in (1): the IMU's accelerometer and gyroscope biases $\mathbf{b}^a, \mathbf{b}^\omega \in \mathbb{R}^3$, respectively. These parameters are modeled as integrated white noise [38], described by the following dynamics:

$$\begin{aligned} \dot{\mathbf{b}}^a &= \eta_{\mathbf{b}^a} \\ \dot{\mathbf{b}}^\omega &= \eta_{\mathbf{b}^\omega} \end{aligned} \tag{2}$$

where $\eta_{\mathbf{b}^a}, \eta_{\mathbf{b}^\omega}$ represent the variance of the zero-mean Gaussian distributions that model the accelerometer and gyroscope drift, respectively. A small variance will denote a highly stable, low-drift IMU.

It is worth mentioning that the LiDAR model includes intrinsic parameters that model the offset of the horizontal and vertical angles of the lasers, as well as the range data [39]. However, they will not be considered within the scope of this work.

As per the extrinsic parameters, the homogeneous transformation matrix that describes the mapping between the IMU and LiDAR local frames is also contemplated in the calibration. In practice, this kinematic relationship is necessary to refer different sensor measurements to the same frame [38].

A rigid transformation ${}^A\mathbf{T}_{AB} \in \mathcal{SE}(3)$ is characterized by a rotation $\mathbf{R}_e \in \mathcal{SO}(3)$ and a translation $\mathbf{t}_e \in \mathbb{R}^3$ from frame A to frame B, defined with respect to frame A. While the latter can be easily represented using a component per spatial axis in the reference frame, the former admits multiple parameterizations (such as Euler angles, quaternions, rotation matrices, ...), each of them with associated benefits and drawbacks [40].

The expression below conforms the most common expression of a rigid motion transformation matrix in a three-dimensional space, adopting a rotation matrix parameterization of $\mathbf{R}_e$.

$$ {}^{A}\mathbf{T}_{AB} = \begin{bmatrix} \mathbf{R}_e & \mathbf{t}_e \\ 0 & 1 \end{bmatrix} \tag{3} $$

The particular formulation of a batch optimization problem that will enable to find the values of the intrinsic and extrinsic parameters presented here (or in other words, the calibration procedure) will be discussed in detail in section 3, along with the necessary additional theoretical background.

2.5 Probabilistic state estimation

From the most general perspective, a model is an abstraction of a real system characterized by an input-output signature. At its core, this behavior is described by a series of equations or *process*, which may be classified according to various criteria (linear or non-linear, dynamic or static, deterministic or stocastic, time variant or invariant, ...) [41]. Regardless of this classification, all processes share the concept of *states*: given an input signal in a certain time instant, states are the variables that enable determining the output of the system.

Note that, in general, there may be more than one set of variables that can achieve this purpose. In other words, the states used to model the input-output behavior of the plant may be selected, and conform what is known as the *state vector*. In practice, some of them may not be directly measurable or may be affected by disturbances; thus, a block known in control theory as a state estimator or *observer* is employed to predict their value from measurements of the input signal and the observable outputs of the real plant.

Within the scope of this project, robot navigation and self-localization can be classified as a time-varying, dynamic process [42]. As briefly discussed in the calibration subsection 2.4, the system parameters may change during runtime and the state vector may include differential equations connecting its variables.

The process or motion model that characterizes the evolution of the robotic platform states is generally non-linear and stochastic, in order to fully grasp the effects of disturbances like wheel slippage and the uncertainty present in the observation model. The latter establishes the relationships that exist between the measurements that the sensors acquire and the state vector.

Finally, to track the evolution of the robot states both the process and observation model are usually embedded into a *filter*. A filter is nothing else than an observer that automatically tunes a gain parameter according to the uncertainty that characterizes states and observations. Such algorithms can be more or less complex (depending on the complexity of the process and observation models themselves); however, note that it is possible to linearize the aforementioned subsystems and still obtain an adequate performance [43].

The particular formulation of the sequential filtering algorithm developed in this dissertation, as well as the choice of state vector, will be discussed in detail in section 4, along with the necessary additional theoretical background.

2.6 State of the art

2.6.1 Intrinsic and extrinsic calibration

As one can intuitively derive, the calibration of multi-sensor systems is hardware-specific. Not only are intrinsic parameters particular to each sensor (as the name implies), the mathematical formulation of the calibration problem depends on the type of data that these devices are able to gather.

Proprioceptive sensors like the IMU are very popular in sensor fusion due to their ubiquitousness and independence with respect to the environment in order to provide measurements. For this reason, their calibration has been addressed in numerous occasions [44], including multi-IMU and camera-inertial configurations [45]. Usually, the latter require the use of artificial calibration targets, which condition the accuracy of the results to the precision with which they are built.

More recently, the LiDAR sensor and the 3D point cloud data it acquires have received greater attention from the *Simultaneous Localization and Mapping* (SLAM) and computer vision communities [21; 24; 46; 32]. Its applications in tandem with monocular and stereo cameras in fields like semantic segmentation and classification [47] or localization and mapping [48] has motivated the design of camera-LiDAR calibration pipelines [49].

However, despite of the popularity of both IMU's and LiDAR's, little work has been addressed towards robustly calibrating LiDAR-inertial systems. The few publicly available methods [50] rely on restrictive a priori conditions, such as the description of a known trajectory. The approach followed by Le Gentil et. al. [38] relaxes these assumptions and opens a path towards target-less calibration of LiDAR-inertial system, inspiring the work discussed in this dissertation.

2.6.2 Probabilistic state estimation

Many proprioceptive and exteroceptive sensors, being *Global Positioning System* (GPS), inertial measurement units, cameras (monocular and stereo) and LiDAR's the most common, have proved their usefulness in state estimation [51; 52]. However, their standalone operation is afflicted by various sources of inaccuracies (e.g. high frequency noise and systematic drift integration in IMU's, false feature correspondence in visual systems, dynamic elements in LiDAR point clouds ...).

For this reason, localization and mapping applications rely on sensor fusion. Regardless of the calibration problem, which is discussed in subsection 2.4, performing data association from multi-modal data sources is significantly more complex [53]. Batch optimization and filtering are the most extended tools used to solve these problems.

Batch optimization methods estimate the states in a local map (i.e. over an interval of time) by solving a large non-linear optimization problem. Fixed-lag smoothers consider constant-sized windows of samples, marginalizing older states e.g. [46], while full smoothing systems consider the whole history of states e.g. [18]. They have been shown to perform well in visual-inertial, visual-LiDAR and LiDAR-inertial configurations [33; 53; 54].

Filtering algorithms, in contrast to batch optimization, perform estimation based solely on the last state; one could said they are *unitary size fixed-lag smoothers*. The most common are those

derived from the *Kalman Filter* [55; 56] due to their easier formulation and flexibility to fuse data; their popularity is remarkably high [57; 58; 59].

Error State Kalman Filters (ESKF) are specially interesting because they work around model linearization to provide optimal filtering for non-linear systems, by indirectly filtering the error in the states rather than their nominal value [60]. In this dissertation, their use is extended to LiDAR-inertial systems by leveraging point cloud matching to compute odometry and correct the drift inherent in inertial-based state propagation.

CHAPTER 3

Offline IMU-LiDAR calibration

Calibration is most usually *not* the ultimate goal when working with a robotic platform. However, it is essential to understand that in order to ensure the best accuracy possible on the upper layers of the robotic system (localization, mapping, path and motion planning, ...) correct calibration of the equipment is a must.

In this section, an IMU-LIDAR offline batch calibration framework heavily inspired by the work of Le Gentil et. at. [38] and Forster et. al. [18] will be developed. It acknowledges the *progressive scanning* nature of the most common LiDAR models, as described in section 2.2, and instead of processing a whole 360° scan as a single instance (i.e. like a global shutter camera would take an image), it subdivides the point clouds in timestamped sectors. The new data format enables dealing with the *motion distortion* phenomenon, which relaxes the assumption that the sensor has remained static during data capture.

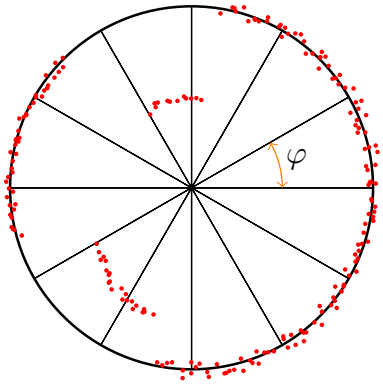

Figure 4: Zenital view of the subdivision of 3D LiDAR 360° scans into φ-wide angular sectors. The angle φ can be modulated by modifying the device's driver.

However, this modification results in a very significant increment of the data rate, exceeding that of the IMU and thus arising the need of upsampling its measurements [38]. The attractiveness of Gaussian Processes (GP) as stochastic interpolators is then threefold: firstly, they enable the characterization of the tendency of the IMU data as a continuous-time function, addressing the sampling issue; secondly, they are able to filter out high frequency noise affecting inertial data, which is a common occurrence and has remarkable impact in state estimation; lastly, it is also able to address data synchronization (commented in section 2.3.2), as it allows to parameterize the temporal shift between the sensors as an optimizable factor.

The LiDAR data is used to estimate and model a map, consisting of an arbitrary set of dominant planes in the scene, and calibrate against it [38]. In this work, the considered calibration map is composed of three planes which are orthogonal to each other. In practice, this is a weak assumption to make, since a convex corner is an adequate target (e.g. the ground and wall planes meet the orthogonality requirements). This is the minimum amount of information necessary to allow full observability of three-dimensional motions.

This information is incorporated in the optimization problem by associating LiDAR points to one of the map planes, and computing the point-to-plane distance as a residual term. Along with IMU pre-integration theory [18] to model the robot motion model, this work formulates a joint calibration and localization problem that only requires the initial pose and velocity of the robot and the calibration map as prior knowledge. This approach is aligned with the objectives set at the beginning of this dissertation (section 1.1). The next sections will delve into the theoretical concepts behind it and the details of its implementation.

3.1 IMU pre-integration

Using the data sampled by the inertial measurement unit (1) it is possible to estimate the robot position $p_k \in \mathbb{R}^3$, velocity $v_k \in \mathbb{R}^3$ and orientation $R_k \in \mathcal{SO}(3)$ at a discrete time instant k by integrating the corresponding states at the previously sampled instant $\{p_{k-1}, v_{k-1}, R_{k-1}\}$ during the $\Delta t_{k-1}^k = t_k - t_{k-1}$ interval. Note that if the integration interval comprises multiple IMU measurements, the implied computational cost increases significantly [18].

However, according to pre-integration theory, it is possible to deal with the aggregation of IMU motion factors between two keyframes into a single constraint [18; 54]. This is attractive for use in factor graphs for full smoothing applications such as the calibration problem at hand, where the whole history of states is estimated by solving a large non-linear problem [18] (more details on section 3.3).

Let (α, β) be two time instants meeting $t_\alpha < t_\beta$, the IMU pre-integration in the $[\alpha, \beta]$ interval can be computed as:

$$
\begin{aligned}
\Delta \mathbf{p}_\beta^\alpha &= \sum_{k=\alpha}^{\beta-1} \Delta \mathbf{v}_k^\alpha \Delta t_{k+1}^k + \mathbf{R}_k^\alpha \left(\mathbf{a}_k - \mathbf{b}^a - \eta^a\right) \left(\Delta t_{k+1}^k\right)^2 \\
\Delta \mathbf{v}_\beta^\alpha &= \sum_{k=\alpha}^{\beta-1} \mathbf{R}_k^\alpha \left(\mathbf{a}_k - \mathbf{b}^a - \eta^a\right) \Delta t_{k+1}^k \\
\Delta \mathbf{R}_\beta^\alpha &= \prod_{k=\alpha}^{\beta-1} \mathrm{Exp}\left(\left(\omega_k - \mathbf{b}^\omega - \eta^\omega\right) \Delta t_{k+1}^k\right)
\end{aligned}
\tag{4}
$$

Where $\{\Delta \mathbf{p}_\beta^\alpha, \Delta \mathbf{v}_\beta^\alpha, \Delta \mathbf{R}_\beta^\alpha\}$ represent the position, velocity and rotation pre-integrated measurements in the $[\alpha, \beta]$ interval, the accelerometer b^a and gyroscope b^ω biases are assumed constant and are affected by Gaussian white noise η^a and η^ω respectively, and $\Delta t_{k+1}^k = t_{k+1} - t_k$ represents the time elapsed between two subsequent measurements, $\{k, k+1\} \in [\alpha, \beta]$.

Note that the linear acceleration and the angular velocity are referred to the global frame. Furthermore, in general, IMU and LiDAR data are *not* synchronized; in order to compute the inertial

measurements at LiDAR timestamps, a time shift parameter δt that models the offset between both sensors has to be introduced.

$$\begin{aligned} \mathbf{a}_k &= \mathbf{a}(t_k - \delta t) \\ \omega_k &= \omega(t_k - \delta t) \end{aligned} \tag{5}$$

3.1.1 Keyframes

The power of pre-integration is related to the computational load relaxation for high volumes of data. It is closely tied to the concept of *keyframes*: instead of integrating all available observations subsequently, these are bundled between some pre-established frames or keyframes (e.g. every full 360° LiDAR spin), resulting in a single motion constraint after applying pre-integration. Thus, the IMU residuals or factors for the optimization problem are only evaluated on these keyframes.

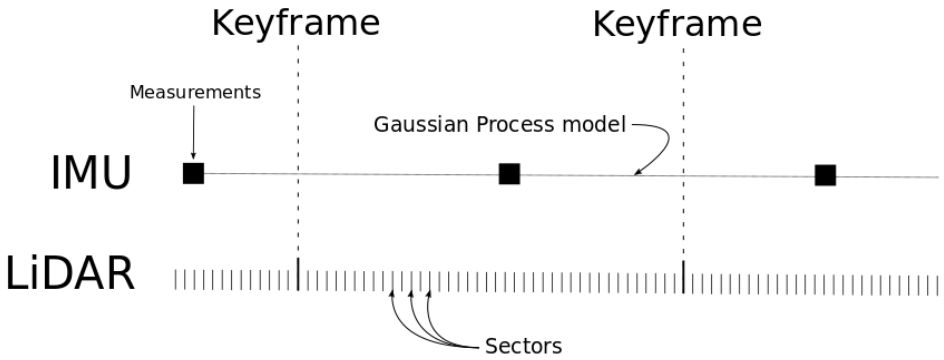

Figure 5: Continuous time modeling of IMU measurements through Gaussian Processes enables computing the robot states at each LiDAR sector. Pre-integration allows summarizing this information in a single motion constraint between keyframes.

Note that (5) assumes that IMU data is well-defined in the continuous domain $\forall t \in \mathbb{R}$; since Gaussian Processes will be used to model the inertial data (more detail in section 3.2), this condition is met. Then, assuming that the robot states referred to the global frame at the initial instant $\{\mathbf{p}_W^\alpha, \mathbf{v}_W^\alpha, \mathbf{R}_W^\alpha\}$ are known, the pose and velocity of the robot at a time t_β, when a certain LiDAR point was captured, $\{\mathbf{p}_W^\beta, \mathbf{v}_W^\beta, \mathbf{R}_W^\beta\}$ are computed as:

$$\begin{aligned} \mathbf{p}_W^\beta &= \mathbf{p}_W^\alpha + \Delta t_\beta^\alpha \mathbf{v}_W^\alpha + \frac{1}{2}\left(\Delta t_\beta^\alpha\right)^2 \mathbf{g}_W + \mathbf{R}_W^\alpha \Delta \mathbf{p}_\beta^\alpha \\ \mathbf{v}_W^\beta &= \mathbf{v}_W^\alpha + \Delta t_\beta^\alpha \mathbf{g}_W + \mathbf{R}_W^\alpha \Delta \mathbf{v}_\beta^\alpha \\ \mathbf{R}_W^\beta &= \mathbf{R}_W^\alpha \Delta \mathbf{R}_\beta^\alpha \end{aligned} \tag{6}$$

3.2 Gaussian Processes

Gaussian Processes (GP) are the generalization of Gaussian probability distributions to functions, which are governed by *stochastic processes* (hence the name) [61]. They are consistent, non-parametric models from the Bayesian Decision theory framework that combine prior knowledge and experimental data to find a posterior distribution over a function, making them suitable for inference, regression and classification tasks.

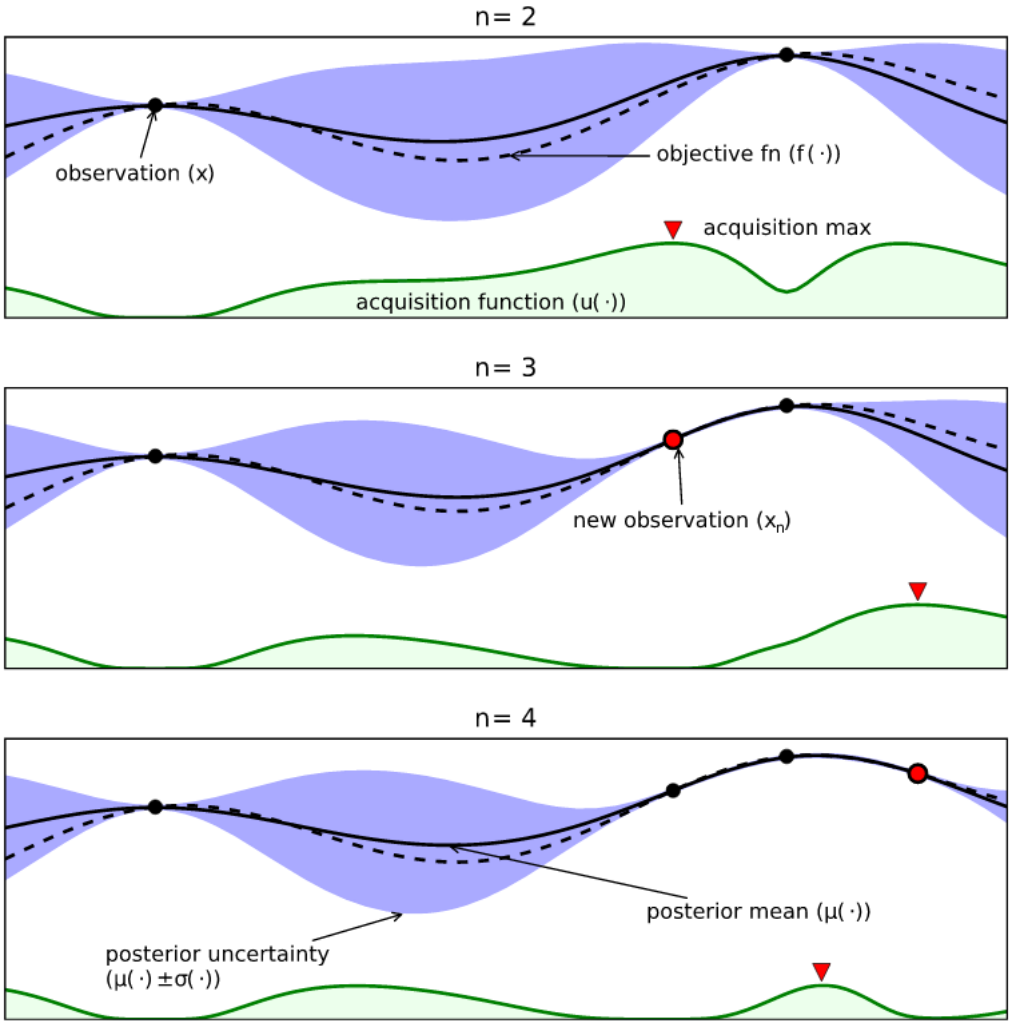

Figure 6: A Gaussian Process model is trained using observations (solid black dots) from a known objective function (dashed black line). The solid black line depicts its mean value, while the grey region around it represents the function covariance. The acquisition function (solid green line) peaks where the model predicts high objectives (exploitation, uncertainty reduction) and high uncertainty (exploration) [62].

Gaussian Processes have received significant attention from the Machine Learning (ML) and the Robotics and Control communities, which overlap in the field of Reinforcement Learning (RL) for decision-based, data-driven systems like controllers [63] or path and motion planners [64; 65]. They have also been successfully introduced in applications like recommendation engines [66] or financial analysis [67], amongst others.

The key factors that describe a Gaussian Process model are the specification of the prior, which fixes properties of the underlying functions (mean, point-wise variance, ...), and most importantly, the selection of covariance function [61]. The latter encodes information about function traits like smoothness or length scale (the faster a function varies, the shorter its length-scale is), which can be interpreted. This inevitably leads to the Bayesian Model selection problem, where the Gaussian Process hyperparameters are tuned for a particular application [62].

Let x be the input data (which is not restricted to be uni-dimensional $\mathbf{x} \in \mathbb{R}^n, n \in \mathbb{N}$), a Gaussian

Process model can be completely specified by the mean $m(x)$ and covariance or kernel function $k(x, x')$ of a real function $f(x)$ [61]:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[\big(f(\mathbf{x}) - m(\mathbf{x})\big)\,\big(f(\mathbf{x}') - m(\mathbf{x}')\big)] \\ f(\mathbf{x}) &\sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \end{aligned} \tag{7}$$

As a result, one can interpret a Gaussian Process as a *collection of Gaussian variables*, i.e. for a certain input x_1, the output of the model is represented by a Gaussian distribution $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$. Note that, as observed in Figure 6, the function variance decreases around the points used to train the model.

The main drawback of Gaussian Processes in practical applications is their poor scalability. Indeed, the basic complexity of operations is $\mathcal{O}(n^3)$ [61], where n is the size of the training set used to learn the model parameters. This becomes prohibitive for large datasets, arising the need of adopting sparsification approaches [68]: selecting only a certain subset as the training samples or *inducing points* can effectively reduce the time complexity for model learning and inference tasks.

3.2.1 Training Gaussian Processes on IMU data

The use case of Gaussian Processes in the formulation of the calibration problem at hand consists in modeling the inertial data. Let $\mathbf{t} \in \mathbb{R}$ be a certain time instant and $\mathbf{a}_t \in \mathbb{R}^3, \omega_t \in \mathbb{R}^3$ the corresponding accelerometer and gyroscope outputs, respectively; six independent Gaussian Process models will be trained using recorded data to represent the IMU measurements in a time interval $[t_i, t_f]$.

$$\begin{cases} a_x(\mathbf{t}) \sim \mathcal{GP}\left(m_{a_x}(\mathbf{t}), k_{a_x}(\mathbf{t}, \mathbf{t}')\right) \\ a_y(\mathbf{t}) \sim \mathcal{GP}\left(m_{a_y}(\mathbf{t}), k_{a_y}(\mathbf{t}, \mathbf{t}')\right) \\ a_z(\mathbf{t}) \sim \mathcal{GP}\left(m_{a_z}(\mathbf{t}), k_{a_z}(\mathbf{t}, \mathbf{t}')\right) \\ \\ \omega_x(\mathbf{t}) \sim \mathcal{GP}\left(m_{\omega_x}(\mathbf{t}), k_{\omega_x}(\mathbf{t}, \mathbf{t}')\right) \\ \omega_y(\mathbf{t}) \sim \mathcal{GP}\left(m_{\omega_y}(\mathbf{t}), k_{\omega_y}(\mathbf{t}, \mathbf{t}')\right) \\ \omega_z(\mathbf{t}) \sim \mathcal{GP}\left(m_{\omega_z}(\mathbf{t}), k_{\omega_z}(\mathbf{t}, \mathbf{t}')\right) \end{cases} \tag{8}$$

Among all the possible choices of covariance function (refer to Chapter 4 of [61] for a more in-depth discussion), the choice for the current setup is the *squared exponential* (SE), which corresponds to a Bayesian linear regression model of an infinite number of basis functions. Its kernel function can be expressed as follows:

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2l^2}\right) \tag{9}$$

Where l is the length-scale parameter of the basis function. An adequate tuning will enable not only an accurate continuous-time representation of the accelerometer and the gyroscope data, but also filtering out the high-frequency noise that affects the inertial device measurements.

The high frequency nature of the IMU sensor showcases the scalability issues of Gaussian Processes when the number of samples, i.e. instances of accelerometer and gyroscope data, grows. This phenomenon is accentuated the wider the time window selected is, making it unfeasible to train the models over the whole set of recorded data (in terms of processing time).

In order to address this, a *temporal window approach* has been followed: the complete time interval $[t_i, t_j]$ has been subdivided in smaller windows $\{[t_i, t_{i+1}], [t_{i+1}, t_{i+2}], \ldots, [t_{j-1}, t_j]\}$ using a sizing criteria (number of samples within the window, or window size in time units), resulting in a piece-wise Gaussian Process definition of the functions in (8). However, while the time complexity of the resulting models will indeed be decreased, the resulting piece-wise representation may incur in a loss of smoothness of the global function due to the presence of local disturbances or changes of tendency.

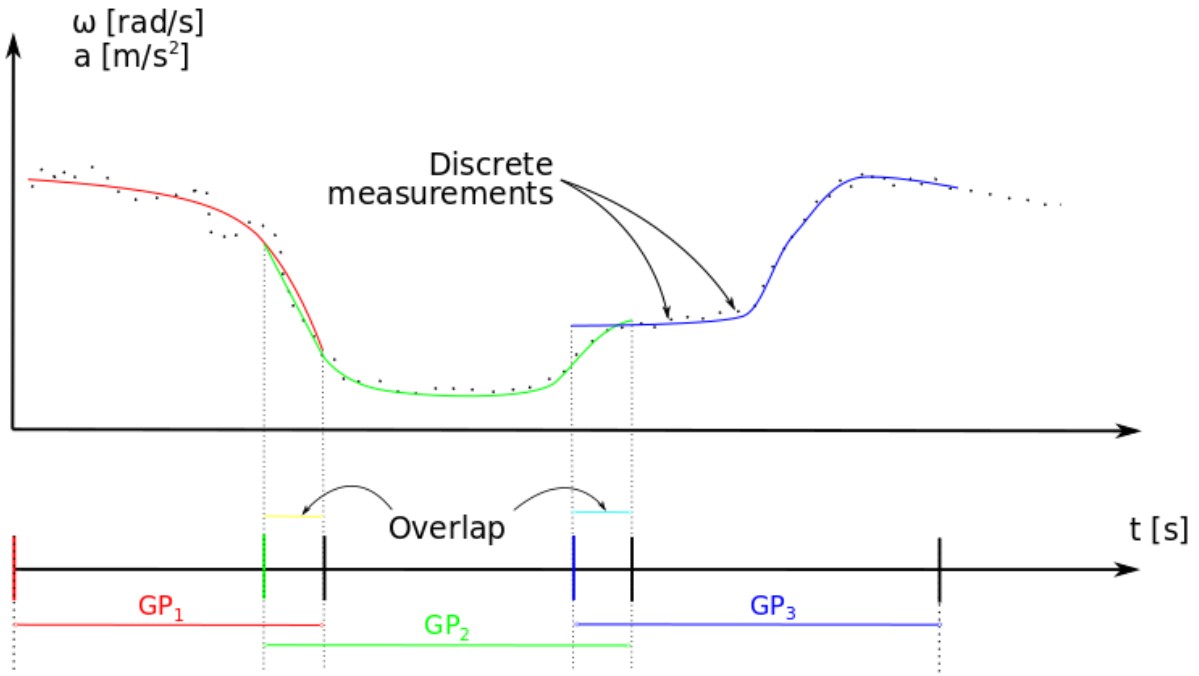

Figure 7: Fitting Gaussian Processes locally saves processing time but compromises function integrity. Overlapping training windows and averaging in those regions strikes a balance between both factors.

The final solution has considered an overlap factor between temporal windows, yielding subsequent Gaussian Process models trained over common time intervals (see Figure 7), aimed to be a trade-off between complexity and smoothness. Additionally, since the inertial data being modeled is involved in an optimization problem (more details in section 3.3), the time complexity of the inference step has been further simplified by generating a dense vector of samples from each temporal window, allowing to perform linear interpolation over that representation (data corresponding to overlapped windows will be averaged accordingly).

3.3 Factor graphs

Factor graphs are a family of probabilistic graphical models that provide an abstraction layer to model and solve large-scale inference problems [69]; many challenges in robotics, including calibration, can be formulated within this framework. One of the most important traits of this representation is its sparse structure, which can be exploited for increased performance.

There exist different notations to express factor graphs [70], albeit in general they consist of a compilation of two types of nodes: factors $f_i \in \mathcal{U}$ and variables $x_j \in \mathcal{V}$, connected via edges $e_{ij} \in \mathcal{E}$. Let $\mathcal{N}(f_i)$ be the adjacency set of f_i, i.e. the set of variables $X_i = \{x_1, \dots, x_k\} \in \mathcal{V}$ that are connected to factor f_i by the set of edges $G = \{e_{i1}, \dots, e_{ik}\} \in \mathcal{E}$, a factor graph $F = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ defines the factorization of the global function $f(X)$:

$$f(X) = \prod_i f_i(X_i) \tag{10}$$

Note that edges encode *independence* relationships between factors and variables as well, since each factor f_i is only a function of its adjacency set $X_i \sim \mathcal{N}(f_i) \in \mathcal{V}$ [69]. This information is valuable in stochastic models, where the inter-dependence between variables is captured by probability distributions.

Then, the IMU-LiDAR calibration problem studied in this section can be formulated in a factor graph framework. The set of variables $S \in \mathcal{V}$ to be optimized is composed by the set of states of the robot at every keyframe, the LiDAR-IMU extrinsic parameters, the IMU biases and the sensor time shift: $S = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_j, \mathbf{R}_e, \mathbf{t}_e, \mathbf{b}^a, \mathbf{b}^\omega, \delta t\}$. A state $\mathbf{s}_m, m \in \{0, 1, \dots, j\}$ denotes a position $\mathbf{p}_m$, velocity $\mathbf{v}_m$ and orientation $\mathbf{R}_m$, as presented in the pre-integration section (3.1). Note that the initial position $\mathbf{p}_0$ must be set in order define the world frame [38]. The adopted approach is graphically represented in Figure 8.

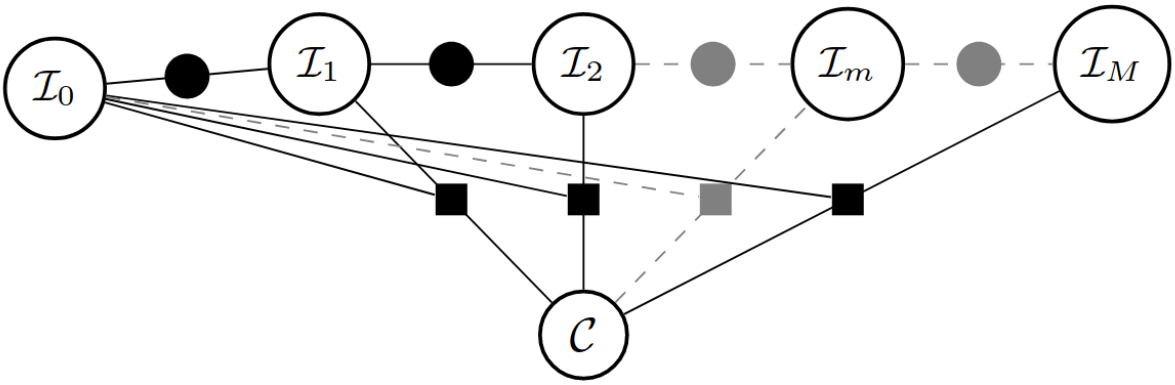

Figure 8: Factor graph that depicts the calibration problem [38]. The IMU nodes $\mathcal{I}_\bullet$ are sequentially connected by IMU factors (black dots), and collectively connected to the calibration set $\mathcal{C}$ through the LiDAR point-to-plane factors.

In Figure 8, $\mathcal{I}_m$ includes the robot states $\{\mathbf{p}_W^m, \mathbf{v}_W^m, \mathbf{R}_W^m\}$ and $\mathcal{C}$ refers to constant parameters, i.e. the extrinsics $\{\mathbf{R}_e, \mathbf{t}_e\}$. Note that the IMU intrinsic biases $\mathbf{b}^a, \mathbf{b}^\omega$ and the temporal shift parameter δt can be considered constant, i.e. $\{\mathbf{b}^a, \mathbf{b}^\omega, \delta t\} \in \mathcal{C}$. However, this work assumes the dynamics of a Wiener process (10) and considers biases and time shift slowly time-varying parameters, arising the necessity of computing correction terms [38] and incorporating them in the states through their Jacobians with respect to these parameters. This is analogous to considering them as part of the robot states $\mathcal{I}_m = \{\mathbf{p}_W^m, \mathbf{v}_W^m, \mathbf{R}_W^m, \mathbf{b}_m^a, \mathbf{b}_m^\omega, \delta t_m\}$.

Let $\bar{\bullet}$ be the prior knowledge of a parameter and $\hat{\bullet}$ be the correction applied after pre-integration, the final value of the parameter after an iteration is $\bullet = \bar{\bullet} + \hat{\bullet}$. The Jacobians of the pre-integration terms in 3.1 with respect to the IMU intrinsic biases and temporal shift is modified as:

$$
\begin{aligned}
\Delta\mathbf{p}^{\alpha}_{\beta(\mathbf{b^a},\mathbf{b}^{\omega},\delta\mathbf{t})} &\approx \Delta\mathbf{p}^{\alpha}_{\beta(\bar{\mathbf{b}}^{\mathbf{a}},\bar{\mathbf{b}}^{\omega},\bar{\delta\mathbf{t}})} + \frac{\partial\Delta\mathbf{p}^{\alpha}_{\beta}}{\partial\mathbf{b}^{a}}\hat{\mathbf{b}}^{a} + \frac{\partial\Delta\mathbf{p}^{\alpha}_{\beta}}{\partial\mathbf{b}^{\omega}}\hat{b}^{\omega} + \frac{\partial\Delta\mathbf{p}^{\alpha}_{\beta}}{\partial\delta t}\hat{\delta t} \\
\Delta\mathbf{v}^{\alpha}_{\beta(\mathbf{b^a},\mathbf{b}^{\omega},\delta\mathbf{t})} &\approx \Delta\mathbf{v}^{\alpha}_{\beta(\bar{\mathbf{b}}^{\mathbf{a}},\bar{\mathbf{b}}^{\omega},\bar{\delta\mathbf{t}})} + \frac{\partial\Delta\mathbf{v}^{\alpha}_{\beta}}{\partial\mathbf{b}^{a}}\hat{\mathbf{b}}^{a} + \frac{\partial\Delta\mathbf{v}^{\alpha}_{\beta}}{\partial\mathbf{b}^{\omega}}\hat{\mathbf{b}}^{\omega} + \frac{\partial\Delta\mathbf{v}^{\alpha}_{\beta}}{\partial\delta t}\hat{\delta t} \\
\Delta\mathbf{R}^{\alpha}_{\beta(\mathbf{b}^{\omega},\delta\mathbf{t})} &\approx \Delta\mathbf{R}^{\alpha}_{\beta(\bar{\mathbf{b}}^{\omega},\bar{\delta\mathbf{t}})} \exp\left(\frac{\partial\Delta\mathbf{R}^{\alpha}_{\beta}}{\partial\mathbf{b}^{\omega}}\hat{\mathbf{b}}^{\omega} + \frac{\partial\Delta\mathbf{R}^{\alpha}_{\beta}}{\partial\delta t}\hat{\delta t}\right)
\end{aligned}
\tag{11}
$$

On the other hand, the factors are related to IMU constraints and reprojection errors of LiDAR points into the calibration map. Since the formulation derived in section 3.1 assumes constant parameters $(\mathbf{b}^a, \mathbf{b}^\omega, \delta t)$, the former accounts for pre-integration errors (which will be corrected via numerically computed Jacobians). The IMU residuals at keyframe $\mathbf{s}_m$ are formulated as follows:

$$
\begin{aligned}
\mathbf{r}^{m}_{p} &= \mathbf{R}^{m\,T}_{W}\left(\mathbf{p}^{m+1}_{W} - \mathbf{p}^{m}_{W} - \Delta t^{m}_{m+1}\mathbf{v}^{m}_{W} - \frac{(\Delta t^{m}_{m+1})^2}{2}\mathbf{g}\right) - \Delta\mathbf{p}^{m}_{m+1} \\
\mathbf{r}^{m}_{v} &= \mathbf{R}^{m\,T}_{W}\left(\mathbf{v}^{m+1}_{W} - \mathbf{v}^{m}_{W} - \Delta t^{m}_{m+1}\,\mathbf{g}\right) - \Delta\mathbf{v}^{m}_{m+1} \\
\mathbf{r}^{m}_{R} &= \log\left(\Delta\mathbf{R}^{m}_{m+1}{}^{T}\mathbf{R}^{m\,T}_{W}\mathbf{R}^{m+1}_{W}\right)
\end{aligned}
\tag{12}
$$

$$
\begin{aligned}
\mathbf{r}^{m}_{\mathbf{b}^a} &= \mathbf{b}^{a}_{m+1} - \mathbf{b}^{a}_{m} \\
\mathbf{r}^{m}_{\mathbf{b}^\omega} &= \mathbf{b}^{\omega}_{m+1} - \mathbf{b}^{\omega}_{m} \\
r^{m}_{\delta t} &= \delta t_{m+1} - \delta t_{m}
\end{aligned}
\tag{13}
$$

$$
\mathbf{r}^{m}_{I} = [\mathbf{r}^{m}_{p}; \mathbf{r}^{m}_{v}; \mathbf{r}^{m}_{R}; \mathbf{r}^{m}_{b^a}; \mathbf{r}^{m}_{b^\omega}] \tag{14}
$$

Reprojection residuals require associating LiDAR points $\mathbf{x}_n = [x_x, x_y, x_z]$ to a plane $\mathbf{P}_i$ within the calibration map $\mathcal{C}$ $(P_i \in \mathcal{C})$, which opens the possibility of establishing point-to-plane distances as factors in the factor graph. This is done by computing the distances d_{ni} from points $\mathbf{x}_n$ to every plane in the calibration map $\mathbf{P}_i \in \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{|\mathcal{C}|}\}$, $d_{ni} \in \{d_{n1}, d_{n2}, \ldots, d_{n|C|}\}$; if the minimum distance is under a certain threshold δ_a, the point will be associated to the corresponding plane.

$$
\mathbf{x}_n \iff \mathbf{P}_i \qquad \text{if} \quad
\begin{cases}
d_{ni} = \min_{d(x_n,\mathcal{C})} \\
d_{ni} < \delta_a
\end{cases}
\tag{15}
$$

For any point captured at any time instant $\mathbf{x}^{L}_{n,t}$, the LiDAR-IMU extrinsic rotation and translation $\{\mathbf{R}_e, \mathbf{t}_e\}$ are used to project it to the IMU frame $\mathbf{x}^{I}_{n,t}$ (16), and from the robot states obtained through pre-integration the point is projected to the world frame, i.e. the initial pose $\mathbf{x}^{W}_{n,0}$ (17). Finally, the point-to-plane distance can be computed after projecting the point to the map frame $\mathbf{x}^{\mathcal{C}}_{n,0}$ (18).

$$\mathbf{x}_{n,t}^{I} = \mathbf{R}_e \mathbf{x}_{n,t}^{L} + \mathbf{t}_e \tag{16}$$

$$\mathbf{x}_{n,0}^{W} = \mathbf{R}_W^t \mathbf{x}_{n,t}^{I} + \mathbf{p}_W^t \tag{17}$$

$$\begin{aligned} \mathbf{x}_{n,0}^{\mathcal{C}} &= \mathbf{R}_e{}^T \left(\mathbf{R}_W^0{}^T \left(\mathbf{x}_{n,0}^{W} - \mathbf{p}_W^0 \right) - \mathbf{t}_e \right) \\ d_n &= \mathbf{n}_i{}^T \mathbf{x}_{n,0}^{\mathcal{C}} + \omega_i \end{aligned} \tag{18}$$

Where $\mathbf{n}_i$ and ω_i are the normal vector to the plane and its distance to the origin, respectively (as formulated in (24)). Note that, irrespective of the calibration map $\mathcal{C}$ size, point-to-plane factors d_n will only be computed with respect to the plane P_i that the point $\mathbf{x}_n$ has been associated to.

Finally, a cost function $F(S)$ that takes into account the point-to-plane and pre-integration factors as residuals, which depend on the robot's state S, is proposed (19). Note that each term is weighted by the inverse of its covariance matrix, that is, the information matrix.

Then, given a set of sensor measurements $\mathcal{Z}$, the optimal set of states S^* that minimize the cost function $F(S)$ can be found as the result of a Maximum Likelihood Estimate problem (MLE) (20):

$$F(S) = \sum_{n=1}^{N} \|d_n\|^2_{Q_{d_n}} + \sum_{m=1}^{M} \|\mathbf{r}_I^m\|^2_{Q_{r_I^m}} + \sum_{m=1}^{M} \left\|r_{\delta t}^m\right\|^2_{Q_{r_{\delta t}^m}} \tag{19}$$

$$S^* = \underset{S}{\operatorname{argmin}}\, F(S) = \underset{S}{\operatorname{argmin}} - \log\left(p(S|\mathcal{Z})\right) \tag{20}$$

In order to solve this large, sparse problem a non-linear optimizer will be employed. A good initial estimate of the states S is required to ensure the convergence of the procedure.

3.3.1 Weighting optimization residuals

In an optimization problem where variables of different orders of magnitude interact, it is important to define adequate weighting functions to balance all the residuals involved in the cost function. This consideration can be extended to the current multi-sensor configuration: unbalanced IMU and LiDAR factors will cause the optimizer to trust one sensor over the other [38], leading to poor calibration results.

In order to coherently size the residuals, the covariance of each measurement $\mathbf{Q}_\bullet$ will be used as a weighting function; or more precisely its inverse form, the *information matrix* $\mathbf{\Delta}_\bullet = \mathbf{Q}_\bullet^{-1}$. This choice of weight will associate a higher cost to a low variance factor, and viceversa; this is reasonable, since the optimizer should penalize harder the more certain (or less uncertain) a residual computation is. The corresponding notation is achieved by expanding (19):

$$F(S) = \sum_{n=1}^{N} \mathbf{\Delta}_{d_n} \|d_n\|^2 + \sum_{m=1}^{M} \mathbf{\Delta}_{\mathbf{r}_I^m} \|\mathbf{r}_I^m\|^2 + \sigma_{\delta t}^{-2} \sum_{m=1}^{M} \left\|r_{\delta t}^m\right\|^2 \tag{21}$$

Note that for the time shift parameter δt the inverse of the covariance $\sigma_{\delta t}^{-2}$ can be taken out of the summatory because it is a constant term. However, one can intuitively see that this is

not the same case for the LiDAR and IMU residuals: the noise accumulated by the sensors' measurements causes the uncertainty (and thus, the covariance magnitude) to grow over time.

Assuming the dynamic model of the error states of the pre-integration measurements (which is closely related to the filtering formulation in the state estimation chapter 4.3), one can recursively update the covariance associated to the residuals [71]. Defining the dynamics $\mathbf{F}_t(t)$ and noise $\mathbf{G}_t(t)$ matrices of the linear system as in (22), the covariance at a certain instant $\mathbf{Q}_{t+\Delta t}$ can be propagated from its value at time Q_t as in (23).

$$\begin{aligned}
\Delta\mathbf{x}_t &= \mathbf{F}_t(t)\Delta\mathbf{z}_t + \mathbf{G}_t(t)\mathbf{n}_t, \qquad \mathbf{n}_t = [n_{a_t}, n_{\omega_t}, n_{b_t^a}, n_{b_t^\omega}]^T \\
\mathbf{F}_t(t) &= \begin{bmatrix} \mathbf{0}_3 & \mathbf{I_3} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{R_t}[\mathbf{a_t} - \mathbf{b_t^a}]_\mathbf{x} & -\mathbf{R_t} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & -[\boldsymbol{\omega}_\mathbf{t} - \mathbf{b_t^\omega}]_\mathbf{x} & \mathbf{0}_3 & \mathbf{I_3} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \\
\mathbf{G}_t(t) &= \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -\mathbf{R_t} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -\mathbf{I_3} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I_3} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I_3} \end{bmatrix}
\end{aligned} \tag{22}$$

$$\begin{aligned}
\mathbf{Q}_{t+\Delta t} &= (\mathbf{I}_{15} + \mathbf{F}_t\Delta t)\,\mathbf{Q}_t\,(\mathbf{I}_{15} + \mathbf{F}_t\Delta t)^T + (\mathbf{G}_t\Delta t)\,\mathbf{W}\,(\mathbf{G}_t\Delta t)^T \\
\mathbf{W} &= \text{diag}\left(\sigma_a^2, \sigma_\omega^2, \sigma_{b^a}^2, \sigma_{b^\omega}^2\right)
\end{aligned} \tag{23}$$

Where $\mathbf{0}_\bullet$ and $\mathbf{I}_\bullet$ are zero and identity matrices of size $\bullet$, respectively; $\mathbf{n}_t$ is the noise vector, which gathers the sources of disturbances in the states (white Gaussian noises corresponding to the accelerometer and gyroscope precision and drift errors, respectively); and $\mathbf{W}$ is the block diagonal noise covariance matrix associated to the noise vector $\mathbf{n}_t$. Indeed, as pointed out by equation (22), the dynamics of the error states constitute a linear, parameter-varying model.

Note that, since the robot pose $\{\mathbf{p}_W^t, \mathbf{R}_W^t\}$ is required to project the LiDAR points to the calibration map, the covariance has to be propagated along the intermediate timestamps between keyframes to fully characterize the uncertainty associated to the point-to-plane factors. The covariance is initialized at zero at the beginning of the calibration $\mathbf{Q}_0 = \mathbf{0}_{15}$ and propagated as per (23).

3.4 Random Sample Consensus (RANSAC)

The Random Sample Consensus technique, widely known as *RANSAC* [72], is a model-fitting algorithm robust to high levels of outliers. The problem this method attempts to solve involves two subproblems that are not independent: classification, or how to find the model that best matches the input data, and parameter estimation, or how to tune the values of the free parameters of the selected model in order to get the best fit possible.

Prior approaches to RANSAC relied on the smoothing assumption, i.e. the idea that regardless of the size of the data set there would always be enough good observations to smooth out the

outliers (e.g. least squares optimization). RANSAC acknowledges that, in general, the smoothing assumption does not hold; in contrast to smoothing techniques, the algorithm initializes a model M_1 using the minimum set of points n necessary to define its free parameters and increasingly builds its *consensus set* S_1, following a hypothesis-test loop [73].

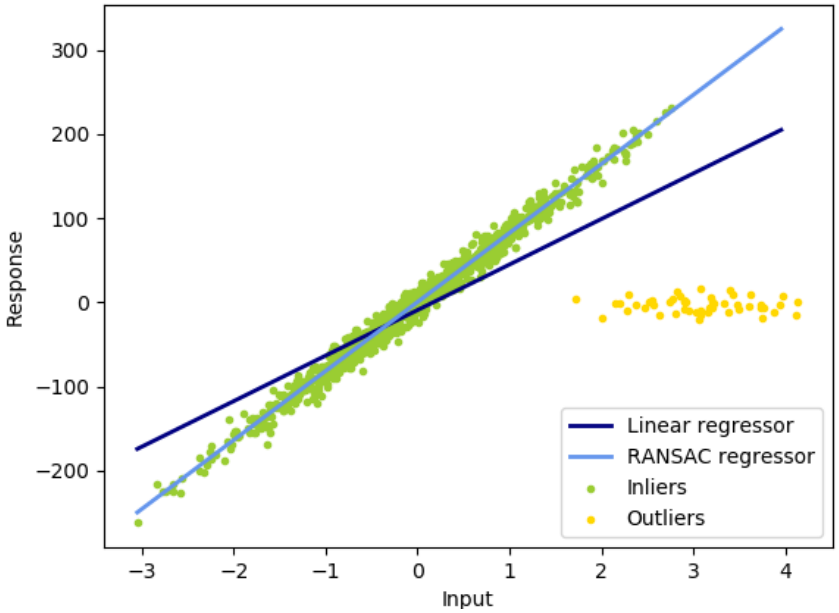

Figure 9: Robust estimation of a line model $f(x) = Ax + b$ using RANSAC (light blue) versus using least squares regression (dark blue) in the presence of outliers (yellow points), i.e. observations that do not comply with the model assumption [74].

This is done by using the current model to determine which other points from the input data pool $P > n$ lie within some error tolerance e. When the size of the consensus set S_1^* exceeds a threshold t (which represents the amount of expected outliers) a new model M_1^* is computed based on said set. If the size of $S_1^* < t$, a new initial subset of points is drawn and the process is repeated. After a certain amount of trials where the consensus set size is smaller than the threshold t, the algorithm terminates; under this paradigm, the model that best fits the data is the one with the highest amount of inliers (i.e. the largest consensus set) [72].

One of the strongest drawbacks of this algorithm is that its maximum computation time is unbounded in principle [75], which is non-compatible with the requirements of real-time applications. Additionally, due of the sampling technique used to select the subsets of points (randomized in its vanilla version), RANSAC may not always yield the same output for the same set of input data. These weaknesses (among others) have been extensively addressed over the years, resulting in numerous variants of the original algorithm [73].

3.4.1 RANSAC-based feature extraction

RANSAC is introduced in this dissertation to obtain planar features from the scene, represented as a three-dimensional point cloud as captured by the LiDAR. As discussed in section 3.3, a set of three orthonormal planes will define the calibration map $\mathcal{C} \sim \{P_0, P_1, P_2\}$. Each of them will be characterized by its parametric model (24):

$$\begin{aligned} &\mathbf{P_i}(\mathbf{p}) \sim A_i p_x + B_i p_y + C_i p_z + D_i = 0\ , \quad i = 0, 1, 2 \\ &\mathbf{p} = [p_x, p_y, p_z]\ , \quad \mathbf{n_i} = [A_i, B_i, C_i]\ , \quad \omega_i = -D_i \end{aligned} \tag{24}$$

Where $\mathbf{n}_i$ is the normal vector perpendicular to the plane, and ω_i is the distance to the origin along that direction. The model tolerance used by the RANSAC algorithm to determine if a point belongs to a plane or not, i.e. it is an inlier or an outlier respectively, is the point-to-plane distance. Let $\mathbf{p}_j \in \mathbb{R}^3$ be a 3D point and $\mathbf{P}_i$ the parametric representation of a plane as in (24), the point-to-plane distance d_{ij} is computed as follows:

$$d_{ij} = \frac{|A_i p_x + B_i p_y + C_i p_z + D_i|}{\sqrt{A_i^2 + B_i^2 + C_i^2}} = \frac{|\mathbf{n_i p_j} - \omega_i|}{\|\mathbf{n_i}\|} \tag{25}$$

Where $|\bullet|$ denotes the absolute value operator, and $\|\bullet\|$ the Euclidean norm of $\bullet$. Note that the point-to-plane distance is also used as a residual in the factor graph (section 3.3). Thus, RANSAC induces a bound to the maximum accuracy achievable during the optimization, given by the error committed on the parameter estimation step.

For the sake of simplifying the actual implementation, some additional assumptions have been made within the scope of this work. First of all, the normal vector of each 3D point towards the viewpoint (i.e. the center of the cloud) is estimated using a nearest-neighbours approach. This information is used to filter out from the cloud points that potentially belong to the *ground plane*, whose normal is assumed to be aligned with the world z axis. The plane equation resulting from applying RANSAC to this set points is added to the calibration map.

Then, the original cloud is filtered again, keeping only the points whose estimated normal is perpendicular to the ground plane (within a certain threshold). No extra assumptions are made in this step, since this condition is imposed by the fact that the planes conforming the calibration map should be orthogonal 3.3. Finally, RANSAC is run on the remaining cloud, only considering as candidates to the map pairs of planes whose normal vectors are perpendiculars with respect to each other.

In order to evaluate the orthogonality between planes, the *cosine similarity* of two normal vectors is computed as:

$$\text{sim}\{\mathbf{P}_p, \mathbf{P}_q\} \equiv s_{pq} = \frac{\mathbf{n}_p \cdot \mathbf{n}_q}{\|\mathbf{n}_p\| \|\mathbf{n}_q\|} \tag{26}$$

Where $\mathbf{n}_p$ and $\mathbf{n}_q$ are the normals associated to planes $\mathbf{P}_p$ and $\mathbf{P}_q$, respectively, and $\mathbf{n}_p \cdot \mathbf{n}_q$ represents the dot product operator. The cosine similarity s_{pq} evaluates to 0 when the vectors considered are perpendicular, as derives from the inner product computation. By choosing a small threshold $\varepsilon \approx 0$, the orthogonality between two planes will be evaluated as:

$$\mathbf{P}_p \perp \mathbf{P}_q \iff |s_{pq}| < \varepsilon \tag{27}$$

Following the procedure above, the resulting map $\mathcal{C}$ will consist of a ground plane $\mathbf{P}_0$ and two *wall* planes $\mathbf{P}_1, \mathbf{P}_2$ (as depicted in Figure 10). Note that this implementation is valid assuming

a z-aligned ground plane exists (which is reasonably common in practice), but it can be generalized further. Either way, the formulation presented and the properties of RANSAC remain invariant to this assumption.

In order to improve the robustness of the *calibration map estimation*, a minimum number of points (inliers) is required to accept a plane as part of the map. Furthermore, if RANSAC is able to fit more than one model to the data, only the dominant plane (i.e. the plane with the maximum amount of inliers) will be kept; this is motivated by the insight that dominant planes will maximize the visibility of the calibration map along the recorded data.

3.5 Capturing data

To obtain a reliable estimate of the intrinsic and extrinsic parameters, the collected data should be *rich enough*. Since grounded robotic platforms (as considered in this dissertation) are *not* able to move freely in space (i.e. their motion is planar), there are non-observable modes of the LTI system underneath the motion model [58].

This suggests that in order to accurately estimate all degrees of freedom of motion, IMU biases and extrinsics it is necessary to excite the sensor rig in such a way that all possible translations and rotations about every axis in the three-dimensional space are covered. The user should be aware of this fact when capturing the data that will be used for calibration.

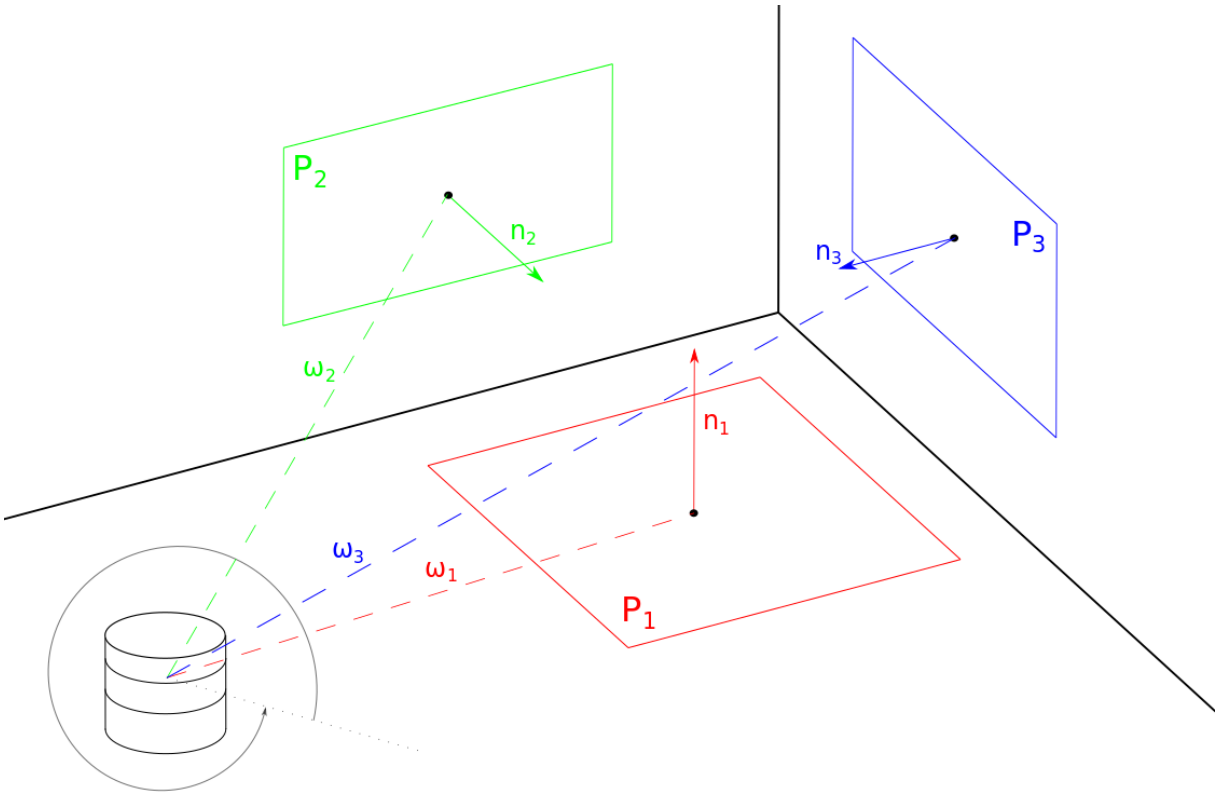

Figure 10: Sketch of a valid calibration map, composed of three orthogonal planes. They are characterized by their normal vector $\mathbf{n}_i$ and their distance to the origin ω_i.

It is also necessary to point out that the calibration map should be visible during the recording, so that the point-to-plane association does not fail. Once again, the user should consider this fact when recording data (i.e. keep the calibration map at a reasonable distance according to the LiDAR's horizontal and vertical field of view), so that IMU and LiDAR factors are balanced. Note that, if it is not possible to associate any point to a certain plane, the corresponding residual in the factor graph will simply be skipped.

3.6 Implementation

The method presented herein has been fully implemented in C++. The *Eigen* [76] library has been chosen to deal with linear algebra including the formulation related to IMU pre-integration, while most point cloud processing utilities can be found in the *Point Cloud Library* [77] project. The latter also features a parallelized version of the RANSAC algorithm, presented in [78], which achieves similar performance but reports a significant processing time improvement. It has been used to extract planar features from LiDAR data, with the objective of building the calibration map and computing point-to-plane distances.

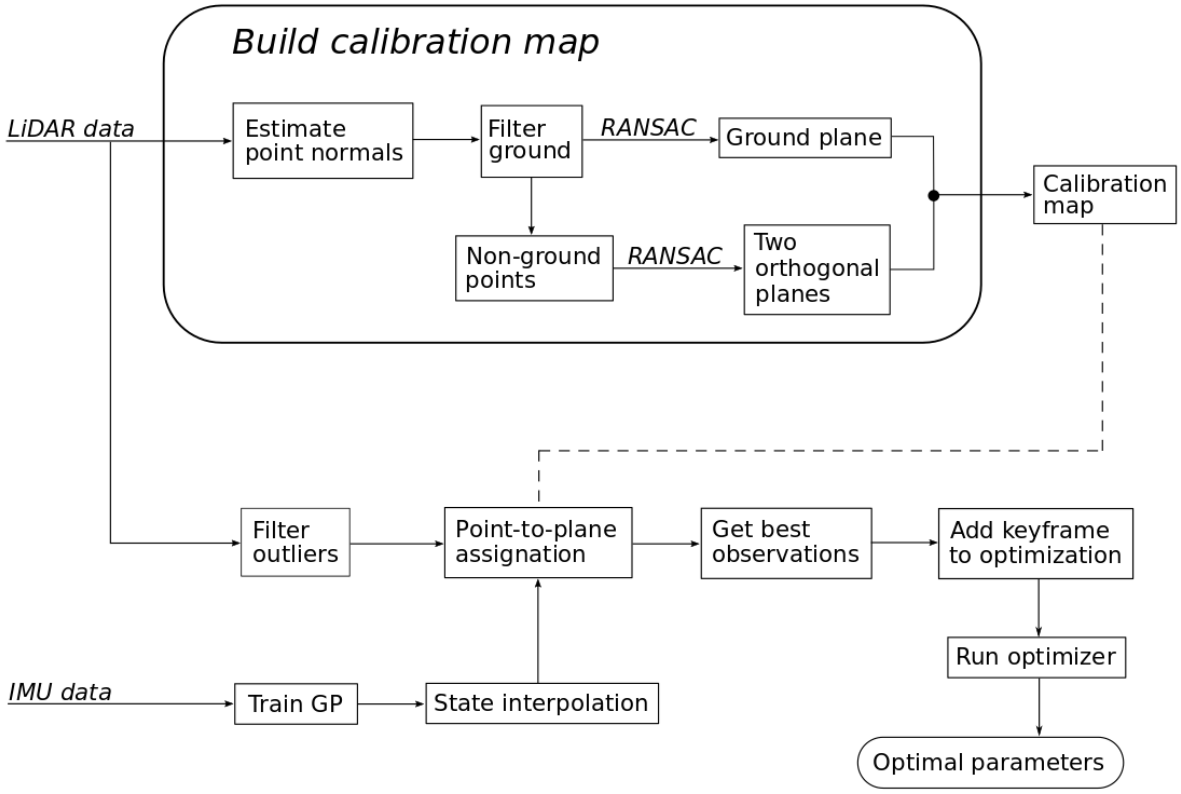

Figure 11: Block diagram of the two-step calibration procedure. Gaussian Processes are trained from IMU data and used to interpolate the robot states at each LiDAR point's timestamp, to perform point-to-plane association (with respect to the calibration map, previously generated from point cloud data).

The *Rosbag* module of the *Robot Operating System* or *ROS* [79], a meta-operating system widely extended in the development of robotic platforms, has been used to record data produced by the IMU and the LiDAR and play it back in an offline setting. The implementation of Gaussian Processes is handled by the *CppGPs* library [80], which provides the necessary functionalities

to generate and train models (including hyperparameter tuning based on marginal likelihood optimization) with square exponential kernels, as well as to perform inference.

Finally, the factor graph implementation and the formulation of the calibration problem as the optimization of a set of parameters has been done in *Ceres* [81], an open-source library for modeling and solving large non-linear optimization problems. The update Jacobians of the pre-integration terms with respect to the parameters have been computed using numerical differentiation, which is available in *Ceres*.

CHAPTER 4

Online multi-sensor state estimation

In order to enable robots to operate autonomously, the first and foremost task (after calibration, that is) is to provide them with the necessary tools for self-localization in the environment [82]. This information is essential for higher level decision making, and thus it should be studied carefully to completely grasp the complexity of stochastic, uncertain, real processes.

This problem is commonly known within the robotics community as *Simultaneous Localization and Mapping* or *SLAM*, which has been investigated in-depth over more than 30 years today [83]. Note that there is a subtle difference between SLAM and odometry: while the latter purely refers to the estimation of an agent's ego-motion via sensor data (such as inertial, visual [84], laser-ranging, ...), the former includes a *loop closure* layer on top [85]. Understanding the topology of the environment through landmark recognition provides defense against wrong data associations and spurious measurements.

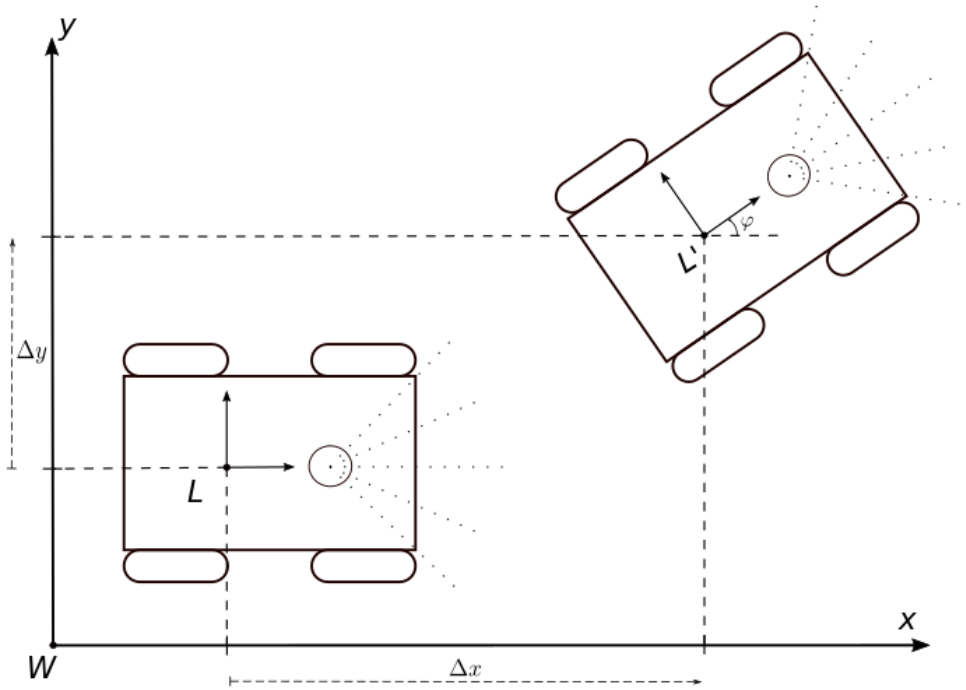

Figure 12: Schematic representation of a 2D odometry problem, where the objective is to find the variation of the nominal states $(\Delta x, \Delta y, \varphi)$ between two frames L and L'.

The work realized in this dissertation, albeit keeping loop closure out of the scope, aims to predict the robot pose (position and orientation) and linear velocity as well as model parameters (i.e. the refined output of the calibration step), accounting for the uncertainty present in real processes, in real time. In other words, the objective is to perform online state estimation in a multi-sensor configuration.

The core idea is to fuse data from an exteroceptive (LiDAR) sensor and a proprioceptive (IMU) sensor to improve state estimation as a result of their complementary nature. Through the combination of the observations provided by the LiDAR (in the form of *scan matching*, which will be covered in section 4.2) and the propagation of inertial data provided by the IMU, it will be possible to update the model parameters online as well. This is desirable, since as discussed in section 3, the IMU intrinsic parameters may drift during operation.

The adopted solution features an *Error State Kalman Filter* (ESKF), an indirect filtering approach that leverages the fact that a nominal, non-linear system may in fact present linear error dynamics which fulfill the Kalman filter optimality condition [57]. The Normal Distributions Transform (NDT) [86; 87] scan matching technique will provide robust pose transform estimates that will enable the computation of an error state observation, used to update the filter and obtain correction factors for the inertial propagation of the nominal states.

The sections in this chapter will delve into the intricacies of working with uncertain 3D poses, which are members of the *special Euclidean* space $\mathcal{SE}(3)$, as well as the filtering and scan matching algorithms briefly introduced above. The code implementation will be detailed at the end of the section.

4.1 Lie theory

Back in the XIX century the mathematician Sophus Lie dedicated most of his research to the study of differential equations and transformation groups. His investigation gave birth to the theoretical foundations of continuous transformation groups, known as *Lie theory*. This area of Mathematics is of great interest to robotics, where the rigorousness and consistency required in the treatment of problems such as state estimation demand an equally appropriate formulation to handle derivates, integrals and uncertainty [88].

As emphasized by several authors [88; 89], Lie theory is significantly complex, albeit for the purposes of this dissertation a limited collection of its core concepts will suffice. Transformations belong to *Lie groups* (also referred to as *manifolds*), which are constructs that merge the concepts of *groups* and *smooth manifolds*. As suggested by the latter, locally they are smooth and resemble a differentiable, linear space: the *Lie algebra*.

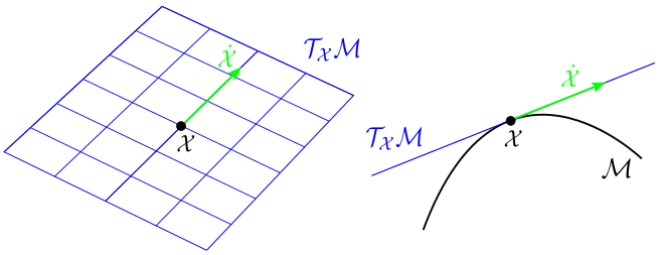

Figure 13: An arbitrary manifold $\mathcal{M}$ and the local tangent space $\mathcal{T}_{\mathcal{X}}\mathcal{M}$ at $\mathcal{X}$. The derivative of a point $\mathcal{X}$, $\dot{\mathcal{X}}$, lives in the tangent space $\mathcal{T}_{\mathcal{X}}\mathcal{M}$, or Lie algebra, at the identity. Drawing inspired by Solà et. al. [88].

One of the main strengths of Lie theory lies in the fact that one can replace a curved, non-linear Lie group $\mathcal{G}$ by its associated Lie algebra $\mathfrak{g}$ [89] or the vector space of $\mathcal{G}$ at the identity, which is generally easier to work with. Thus, the evolution of a given set of states around the identity in the manifold $\mathcal{M}$ can be identified with vectors in the tangent space $\mathfrak{m}$ or *the Lie algebra of* $\mathcal{M}$ in $\mathbb{R}^m$ [88], where m is the amount of degrees of freedom of $\mathcal{M}$.

To stress the importance of this relationship, consider that the *special orthogonal* group $\mathcal{SO}(3)$ of 3D rotation matrices or the *special Euclidean* manifold $\mathcal{SE}(3)$ of 3D rigid motions, which are extensively used in robotics, are cases of Lie groups. Lie theory provides a framework to accurately formulate differential algebra and uncertainty propagation in these groups.

4.1.1 Mapping between spaces

The conversion between members of a Lie group $\mathcal{M}$ and elements of its Lie algebra $\mathfrak{m}$ is exact [88]. The *exponential map* wraps a vector in the tangent space at the identity $\mathfrak{m}$ to an arc over the manifold $\mathcal{M}$, and the *logarithmic map* performs the inverse operation.

$$\begin{aligned} \exp &: \mathfrak{m} \mapsto \mathcal{M} \; ; \quad \tau^\wedge \in \mathfrak{m} \mapsto \mathcal{Z} \in \mathcal{M} \\ \log &: \mathcal{M} \mapsto \mathfrak{m} \; ; \quad \mathcal{Z} \in \mathcal{M} \mapsto \tau^\wedge \in \mathfrak{m} \end{aligned} \tag{28}$$

In turn, any member of the Lie algebra $\mathfrak{m}$ of $\mathcal{M}$ can be expressed as a linear combination of a set of base elements or *generators* $\mathbf{E} \in \mathfrak{m}$. This property provides an invertible mapping or *isomorphism* [88] between the Cartesian vector space $\mathbb{R}^m$ and the Lie algebra $\mathfrak{m}$ and viceversa, indicated by the *wedge* and *vee* operators, respectively.

$$\begin{aligned} \textit{wedge} : \mathbb{R}^m \mapsto \mathfrak{m} \; ; \quad \tau \mapsto \tau^\wedge = \sum_{i=1}^{m} \tau_i \mathbf{E}_i \\ \textit{vee} : \mathfrak{m} \mapsto \mathbb{R}^m \; ; \quad \tau^\wedge \mapsto (\tau^\wedge)^\vee = \tau = \sum_{i=1}^{m} \tau_i \mathbf{e}_i \end{aligned} \tag{29}$$

The aforementioned mappings are summarized in Figure 14.

Finally, vector spaces at an arbitrary point of the manifold $\mathcal{Z} \in \mathcal{M}$ can be mapped to the Lie algebra at the identity via a linear transform, the *adjoint action*. Given the isomorphisims $\tau_\mathcal{Z} \doteq {\tau_\mathcal{Z}}^\wedge$, $\tau_\mathcal{I} \doteq {\tau_\mathcal{I}}^\wedge$, using the subscript $\bullet_\mathcal{I}$ to denote the identity, the adjoint matrix $Adj_\mathcal{Z}$ maps the former Cartesian space to the latter as per (30).

$$Adj_\mathcal{Z} : \mathbb{R}^m \mapsto \mathbb{R}^m \; ; \quad \tau_\mathcal{Z} \mapsto \tau_\mathcal{I} = Adj_\mathcal{Z} \, \tau_\mathcal{Z} \tag{30}$$

Lie algebra of $\mathcal{SO}(3)$ The special orthogonal group $\mathcal{SO}(3)$ consists of three dimensional rotations, denoted by rotation matrices $\mathbf{R} \in \mathcal{M}$ that meet the property $\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$, where $\bullet^T$ indicates matrix transposition and $\mathbf{I}_3$ is the identity matrix. The representation of this group in Cartesian space admits various parameterizations, being *Euler angles* $\theta = \{\phi, \omega, \psi\} \in \mathbb{R}^3$ one of the most common.

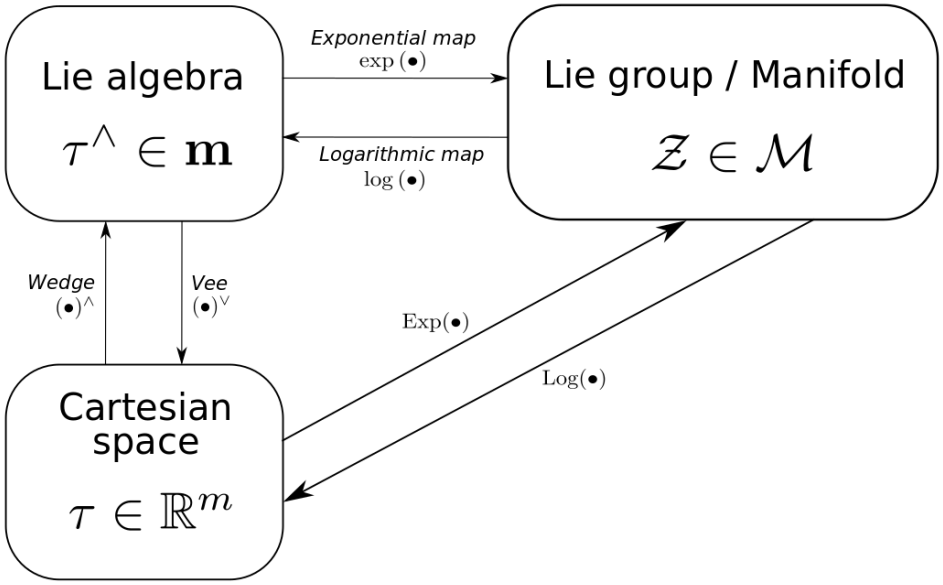

Figure 14: Block diagram of the mappings between different spaces (Cartesian vector space, Lie algebra and Lie group or manifold).

In this Lie group, the $\theta^\wedge$ operator converts a member from the Cartesian space $\theta \in \mathbb{R}^3$ to the space of skew-symmetric matrices $[\theta]_x \in so(3)$, i.e. the Lie algebra of $so(3)$ (31). On the other, $\exp(\theta^\wedge)$ relates the rotation on the manifold $\mathcal{SO}(3)$ with its Lie algebra at the identity $\theta^\wedge \in so(3)$ through Rodrigues' formula [18]. The $\bullet^\vee$ operator performs the inverse mapping with respect to the former, and $\log(\bullet^\wedge)$ to the latter.

$$\begin{aligned}
&\bullet\ \mathbb{R}^3 \mapsto so(3)\ ; \quad \theta \in \mathbb{R}^3 \mapsto \theta^\wedge = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\psi & \omega \\ \psi & 0 & -\phi \\ -\omega & \phi & 0 \end{bmatrix} \\
&\bullet\ so(3) \mapsto \mathcal{SO}(3)\ ; \quad \theta^\wedge \in so(3) \mapsto \exp(\theta^\wedge) = \mathbf{I}_3 + \frac{\sin\|\theta\|}{\|\theta\|}\theta^\wedge + \frac{1-\cos\|\theta\|}{\|\theta\|^2}(\theta^\wedge)^2
\end{aligned} \tag{31}$$

Lie algebra of $\mathcal{SE}(3)$ The special Euclidean group $\mathcal{SE}(3)$ encompasses rigid motions in three dimensional spaces, which consist in a translation and a rotation applied simultaneously [88]. As a result, the dimension of this group is 6, since both translation $\mathbf{t} \in \mathbb{R}^3$ and rotation $R \in \mathcal{SO}(3) \mapsto \theta \in \mathbb{R}^3$ are parameterized by three dimensional vectors each (in Cartesian space).

The special Euclidean manifold is populated by rigid transformation matrices $T \in \mathcal{SE}(3)$, which follow the formulation in (32), where $\mathbf{R} \in \mathcal{SO}(3)$ is the result of applying the exponential map to the Euler angles $\theta \in \mathbb{R}^3 \mapsto \exp(\theta^\wedge) \in \mathcal{SO}(3)$. The associated Lie algebra $se(3)$ is also presented in (32).

$$
\begin{aligned}
&\bullet\ \mathbf{T} \in \mathcal{SE}(3)\ ; \quad \mathbf{t} \in \mathbb{R}^3\ , \quad R \in \mathcal{SO}(3) \mapsto \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \\
&\bullet\ \mathbb{R}^6 \mapsto \mathit{se}(3)\ ; \quad \mathbf{t} \in \mathbb{R}^3\ , \quad \theta \in \mathbb{R}^3 \mapsto \begin{bmatrix} \mathbf{t} \\ \theta \end{bmatrix} \in \mathbb{R}^6 \mapsto \begin{bmatrix} [\theta]_x & \mathbf{t} \\ 0 & 0 \end{bmatrix} \in \mathit{se}(3) \\
&\bullet\ \mathit{se}(3) \mapsto \mathcal{SE}(3)\ ; \quad \exp\left(\begin{bmatrix} [\theta]_x & \mathbf{t} \\ 0 & 0 \end{bmatrix} \right) \in \mathcal{SE}(3)
\end{aligned}
\tag{32}
$$

4.1.2 Uncertainty propagation in $SE(3)$

Now, the focus is put on how to handle uncertainty in 3D rigid motions, i.e. in the Lie algebra of $\mathcal{SE}(3)$, $\mathit{se}(3)$. Its consistent appearance in recent robotics and computer vision problems has motivated specific literature on this topic [90; 91].

Let $\mathbf{T}_{ij}$ be a homogeneous transform matrix, which describes a rotation expressed as a set of Euler angles $\mathbf{R}_{ij} = f(\phi_{ij}, \omega_{ij}, \psi_{ij})$ and a 3D translation $\mathbf{t}_{ij} = [t_{ij}^x, t_{ij}^y, t_{ij}^z]$, that transforms pose i to pose j. Be ξ_{ij} a zero-mean Gaussian random variable $\xi_{ij} \sim \mathcal{N}(0, \mathbf{\Sigma}_{ij}) \in \mathbb{R}^6$, the transform $\mathbf{T}_{ij}$ can be defined as an $\mathcal{SE}(3)$ random variable following [91] notation:

$$
\mathbf{T}_{ij} := \exp\left(\xi_{ij}^{\wedge}\right) \bar{\mathbf{T}}_{ij} \tag{33}
$$

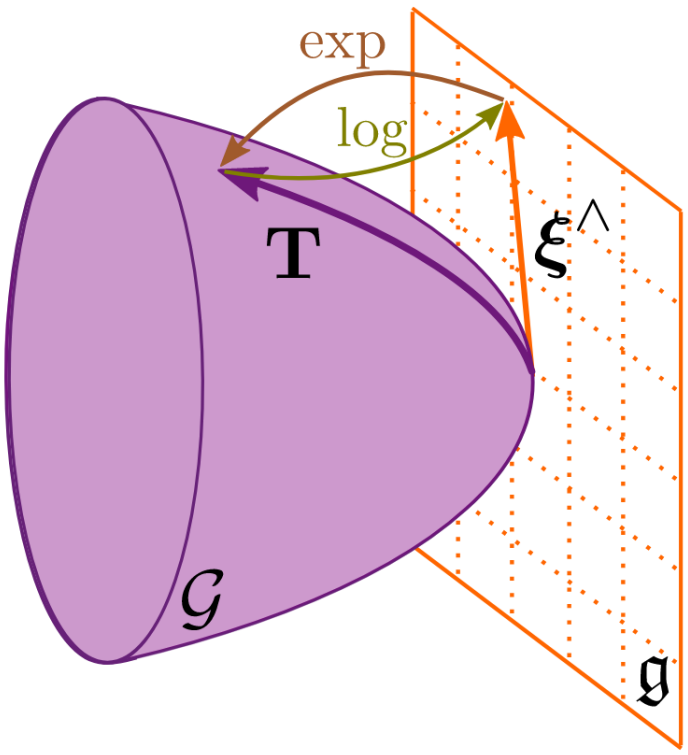

Figure 15: Representation of a rigid transform $\mathbf{T}$ on the manifold $\mathcal{G}$ and its associated Gaussian noise $\xi^{\wedge}$ on the tangent space at the identity $\mathcal{g}$. The exponential and logarithmic mappings between $\mathcal{G}$ and $\mathcal{g}$ are also depicted [91].

Where $\mathbf{T}_{ij}$ becomes a Probability Distribution Function (PDF) over the $\mathcal{SE}(3)$ group, characterized by a mean value $\bar{\mathbf{T}}_{ij}$ and a covariance $\mathbf{\Sigma}_{ij} \in \mathbb{R}^{6x6}$. Now, any element $\xi^{\wedge} \in \mathfrak{se}(3)$ can be represented by a set of 6 generators [92]. Three of them are related to a 3D translation (about x, y and z axes), while the other three represent rotations about the same reference frame axes, i.e. the Euler angles (ϕ, ω, ψ). Then, the isomorphism $\delta \in \mathbb{R}^6$ and the adjoint matrix Adj_T can be noted as in (34).

$$\begin{aligned} \delta &= [t_x \quad t_y \quad t_z \quad \phi \quad \omega \quad \psi]^T = \begin{bmatrix} \mathbf{t} \\ \mathbf{R} \end{bmatrix} \\ \text{Adj}_T &= \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_x \mathbf{R} \\ 0_3 & \mathbf{R} \end{bmatrix} \end{aligned} \tag{34}$$

Where $[\bullet]_x$ represents the skew-symmetric operator (subsection 4.1.1), $\mathbf{t}$ is the translation vector denoted by ξ and $\mathbf{R}$ is the rotation matrix generated by the Euler angles in ξ.

Pose composition Suppose that two uncertain, consecutive poses $\mathbf{T}_{ij}$ and $\mathbf{T}_{jk}$ are known and modeled as in (33), and the goal is to find the end-to-end transform $\mathbf{T}_{ik}$. As detailed in [90; 91], the mean propagation $\bar{\mathbf{T}}_{ik}$ follows the $\mathcal{SE}(3)$ group composition operator.

$$\bar{\mathbf{T}}_{ik} \triangleq \bar{\mathbf{T}}_{ij} \bar{\mathbf{T}}_{jk} \tag{35}$$

As per the covariance, its computation intrinsically depends on the independence assumption between poses $\mathbf{T}_{ij}$ and $\mathbf{T}_{jk}$. Generalizing to the case where the distributions are correlated, a first order estimate of the covariance is obtained as [93]:

$$\begin{aligned} \mathbf{\Sigma}_{ik} \approx \mathbf{\Sigma}_{ij} + \text{Adj}_{\bar{\mathbf{T}}_{ij}} \mathbf{\Sigma}_{jk} \text{Adj}_{\bar{\mathbf{T}}_{ij}}{}^T + \\ \mathbf{\Sigma}_{ij,jk} \text{Adj}_{\bar{\mathbf{T}}_{ij}}{}^T + \text{Adj}_{\bar{\mathbf{T}}_{ij}} \mathbf{\Sigma}_{ij,jk}^T \end{aligned} \tag{36}$$

Where $\text{Adj}_{\bullet}$ denotes the adjoint action on the Lie group of $\bullet$. Note that the terms in which the cross-correlation $\mathbf{\Sigma}_{ij,jk}$ appears are nullified under the hypothesis that the poses $\mathbf{T}_{ij}$ and $\mathbf{T}_{jk}$ are independent (i.e. uncorrelated). This assumption will be made within the scope of this work.

Pose inversion Suppose now an uncertain pose $\mathbf{T}_{ij}$, which denotes the position and orientation of frame j with respect to frame i. The goal is to change the reference frame, that is, to find the position and orientation of frame i with respect to frame j, $\mathbf{T}_{ji}$, which implies the application of the inverse operator on $\mathbf{T}_{ij}$.

The mean $\bar{\mathbf{T}}_{ji}$ and covariance $\mathbf{\Sigma}_{ji}$ of the inverted pose are represented by [93]:

$$\begin{aligned} \bar{\mathbf{T}}_{ji} &= \bar{\mathbf{T}}_{ij}^{-1} \\ \mathbf{\Sigma}_{ji} &= \text{Adj}_{\bar{\mathbf{T}}_{ij}^{-1}} \mathbf{\Sigma}_{ij} \text{Adj}_{\bar{\mathbf{T}}_{ij}^{-1}}{}^T \end{aligned} \tag{37}$$

4.2 Scan matching

Exteroceptive sensors provide features that describe the world around them; as briefly introduced in section 2.2, for the LiDAR this representation consists in a three-dimensional point cloud. As it stands, multiple instances of these representations may depict a common element present in the environment, but from a different point of view. Finding the rigid motion $\mathbf{T} \in \mathcal{SE}(3)$ that transforms one viewpoint frame to the other is known as the problem of *scan matching* or *point cloud registration* [94].

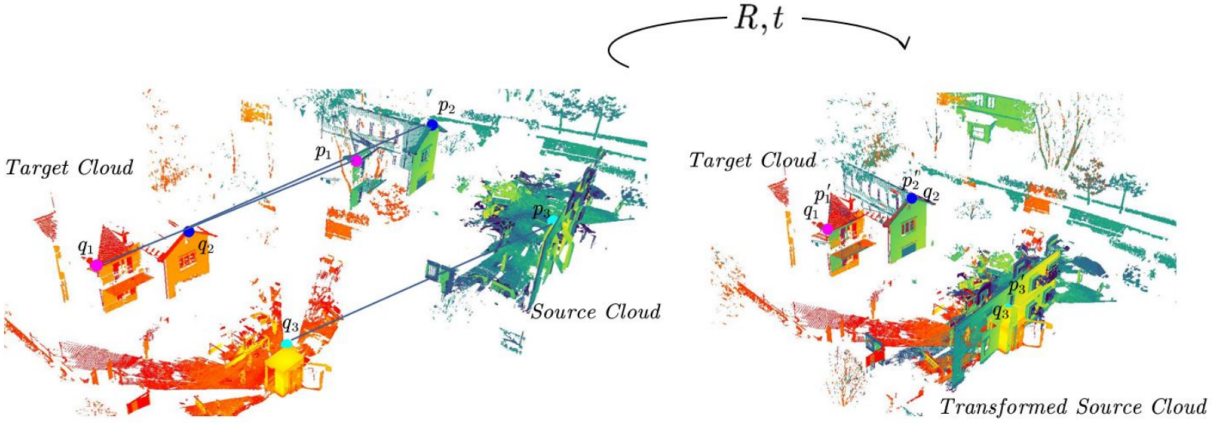

Figure 16: Example of a scan registration, where $p_\bullet$ and $q_\bullet$ are point correspondences (i.e. physical entities seen from different perspectives) between a source point cloud and a target cloud, and the objective is to estimate the transformation parameters $\mathbf{T} \in \mathcal{SE}(3) \mapsto \{\mathbf{t} \in \mathbb{R}^3, \mathbf{R} \in \mathcal{SO}(3)\}$ [94].

Although scan registration can be generalized for multi-view problems, in an homologous way to monocular and stereo image matching (e.g. [95; 96]), for the purpose of this dissertation the scope is set to two-view registration. More exactly, the LiDAR is considered rigidly attached to a body; in this configuration, the rigid motion $\mathbf{T} \in \mathcal{SE}(3)$ estimated by the scan matching technique produces none other than an *odometry* estimate of said body.

This approach to the scan matching problem is also known as *LiDAR odometry*. Purely LiDAR-based odometry was proposed initially Zhang et. al. [52], but it has become a popular source of data fusion with inertial and visual sensors [33; 36; 46]. In general, these methods rely on an initial step of finding correspondences between descriptors in the point clouds (point-to-point, feature-to-feature, point-to-feature, ...)

Note that feature correspondence, i.e. data association, is a complex problem in and of itself, and thus scan matching techniques have to be designed carefully to consider factors like scaling, robustness to outliers, ... [32]. However, if the dependency to explicit correspondences were to be removed, part of the issues inherent to data association (occlusions, spurious noise, ...) could be addressed; for this reason, the *Normal Distributions Transform* or *NDT* by Biber et. al [86] has been explored in this dissertation.

4.2.1 Normal Distributions Transform (NDT)

The Normal Distributions Transform is a scan registration technique that aims to model point clouds as piecewise continuous and differentiable Gaussian distributions [86]. By subdividing the space in constant-sized cells or voxels, the mean and covariance of the normal distribution

that models the probability of finding a sample within a cell is quantified. Thus, cloud matching in this framework involves optimizing a maximum likelihood (or a minimum negative log-likelihood) function via the PDFs' parameters, which yields an estimate of the translation $\hat{\mathbf{t}} \in \mathbb{R}^3$ and rotation $\hat{\theta} \in \mathcal{SO}(3)$ that align a source cloud with a target cloud.

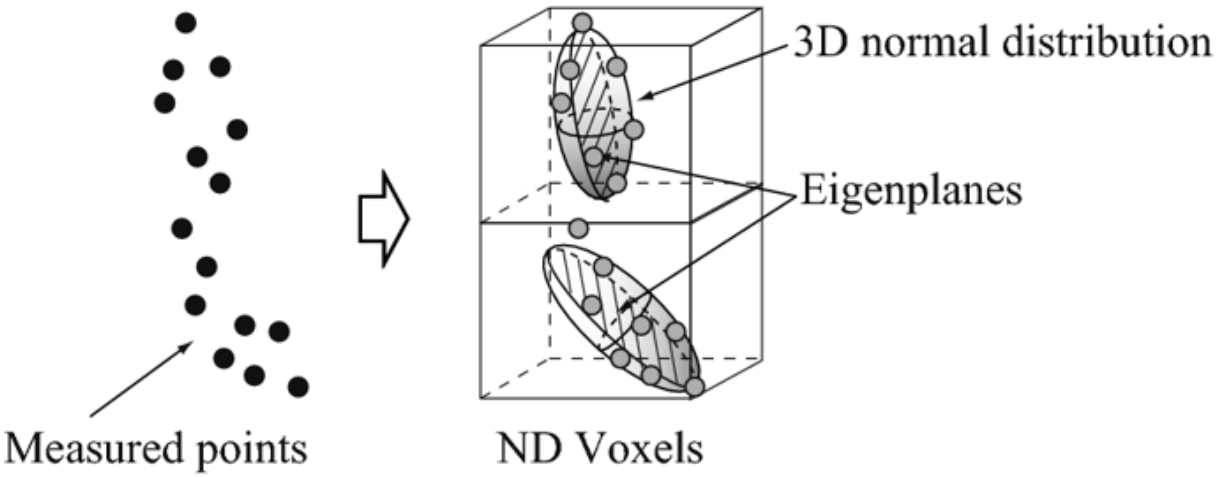

Figure 17: The Normal Distributions Transform (NDT) discretizes the world in 3D voxels and describes the LiDAR point density within each as a Gaussian PDF [97].

Although this technique was initially developed for 2D spaces, Magnusson [87] extended the original NDT formulation to consider 3D point clouds. In his work he acknowledges the fact that the PDF assumed over the scan points is not required to be Gaussian (which allows to embed different distributions to achieve different properties); however, this choice enables the derivation of analytic expressions that define a score function $s(\vec{\mathbf{p}})$, as well as its gradient vector g and Hessian matrix $\mathbf{H}$ with respect to the distribution parameters $\vec{\mathbf{p}}$ (refer to [87] for the explicit formulation).

Consequently, the transform parameters $\vec{\mathbf{p}}$ $\{\mathbf{t} \in \mathbb{R}^3,\ \theta \in \mathbb{R}^3\}$, which describe a rigid motion $\mathbf{T}(\vec{\mathbf{p}}) \in \mathcal{SE}(3)$, that optimize the score function $s(\vec{\mathbf{p}})$ can be found using an iterative Newton's method-based solver in the form $\mathbf{H}\Delta\vec{\mathbf{p}} = -\mathbf{g}$, where $\Delta\vec{\mathbf{p}}$ is an increment to be applied to the initial guess of the parameter vector $\vec{\mathbf{p}} \in \mathbb{R}^6$. As noted by Akai et. al. [31], since this is a gradient-based method, the convergence of this procedure heavily depends on the parameter initialization.

On the positive side, this technique carries low computational cost and enables a probabilistic treatment of the odometry estimates, which aligns extremely well with the online filtering framework. Furthermore, the fusion with inertial data from an IMU can help improve the convergence of the cloud matching, since it provides a prior guess of the rigid motion between scans (based on state propagation, as discussed in 4.3).

Transform covariance estimation Characterizing the uncertainty of an observation is not trivial; in fact, it will impact substantially the performance of a filter, so an effort must be made to estimate the covariance of a measurement accurately. As discussed by the author [87], the Normal Distributions Transform provides the tools to do so.

The score function is a reasonable quality index for a transform, since it is the objective function to be optimized. By applying a $1/n$ scaling factor (38) a metric independent of the amount of

points n in the clouds, $\mathbf{Q}_s$, can be used to compare different matchings. On the other hand, the inverse of the Hessian matrix $\mathbf{H}$ encodes the certainty by which each parameter $p_i \in \vec{\mathbf{p}}$ was determined, lending itself to be used as a covariance estimate $\mathbf{Q}_H$ of said parameters (38).

$$\begin{aligned} \mathbf{Q}_s &= \text{diag}\left(\frac{1}{n} s(\vec{\mathbf{p}})\right) \in \mathbb{R}^{6x6} \\ \mathbf{Q}_H &= \mathbf{H}^{-1} \in \mathbb{R}^{6x6} \end{aligned} \tag{38}$$

In (38), diag(•) indicates a diagonal matrix with • elements in the diagonal. Note that the metric computed through the Hessian inverse provides higher discriminability between the degrees of freedom of the transform, i.e. translation and rotation about the three spatial axes. This information may be of significance for a filtering algorithm, since it is able to describe motions that are not equally observable on all its parameters (e.g. planar motion).

Algorithm hyperparameters The NDT technique relies on some hyperparameters to be set, which are listed below:

- **Voxel size:** The selection of resolution with which to discretize the world heavily depends on the device that captured the data, according to parameters as its field of view or the density of points in the cloud [87]. If it is not adequately set, the structure of the scene may not be captured, resulting in unsuccessful matchings. Note that as a consequence of using a gradient-based solver, the algorithm may converge towards local minimum rather than reporting that a registration has failed. Also, note that larger voxel sizes do not directly imply a more coarse matching (and viceversa), because its quality of the depends on the size of the features present in the scan.

- **Outlier ratio:** In order to minimize the influence of outliers in the objective function, an extra term is added to the assumed Gaussian PDF. It is a uniform distribution of a value proportional to the expected ratio of outlier points, which enables bounding their influence (since the parameters of the estimated distribution must maintain the properties of the PDF) [87].

- **Convergence criteria:** The algorithm assumes it has converged upon reaching a maximum number of iterations, or when the Euclidean norm of the step size (i.e. the magnitude of the variation of the parameter vector $\vec{p}$ between subsequent iterations) is smaller than a certain factor. A trade-off between the processing time and the matching accuracy can be set by tuning these hyperparameters.

Voting system In an attempt to minimize the influence of non-optimal hyperparameters in the transform estimation, a module inspired by the sampling principle of particle filters is designed. It performs multiple NDT's in parallel for k different sets of hyperparameters, which in turn yield a set of transform candidates $\mathcal{T} = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_k\}$, characterized by their corresponding parameter vectors $\{\vec{\mathbf{p}}_1, \vec{\mathbf{p}}_2, \ldots, \vec{\mathbf{p}}_k\}$.

The metric used to evaluate the quality of the matching is the *inverse of the trace of the estimated covariance matrix* (39), which is a good indicator of the uncertainty of the registration even for ill-conditioned matrices (e.g. when a certain degree of freedom is not observable due to planar

motion). A higher value of the metric (i.e. a smaller trace) will indicate that the uncertainty of the matching is lower.

$$\lambda_j = \frac{1}{\text{tr}(\mathbf{Q}_{\mathbf{T}_j})}, \qquad j = \{1, 2, \dots, k\} \tag{39}$$

The voting system then performs statistical outlier removal by computing the first and second moments of the metric (i.e. its mean and covariance $\lambda \sim \mathcal{N}(\mu_\lambda, \sigma_\lambda)$) and eliminating any observation whose trace is at least $3\sigma_\lambda$ over the mean. In other words, the set of outlier transform estimates is defined as $\mathcal{O} = \{\mathbf{T}_j \mid \lambda_j > \mu_\lambda + 3\sigma_\lambda\}$.

From the set of remaining candidates $\mathcal{I} = \mathcal{T} - \mathcal{O}$ of size l, a weighted average is performed. Each observation is assigned a weight w_j as per (40) and assembled into a weight vector $\mathbf{w} = [w_1, w_2, \dots, w_l]$. Translations $\mathbf{t}_j \in \mathbb{R}^3$, rotations $\theta_j \in \mathbb{R}^3 \mapsto \mathbf{q}_j \in \mathcal{S}^3$ and covariances $\mathbf{Q}_j \in Sym_d^+$, which lie in the space of symmetric positive definite matrices Sym_d^+, are decoupled.

$$\begin{aligned} &\mathbf{w} = [w_1, w_2, \dots, w_l], \qquad \mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_l] \\ &w_j = \frac{\lambda_j}{\sum_{i=1}^{l} \lambda_i}, \qquad \sum_{i=1}^{l} w_i = 1 \\ &\rightarrow \mu_t = \mathbf{w}\mathbf{t} \end{aligned} \tag{40}$$

The mean translation $\mu_t \in \mathbb{R}^3$ is easily computed (40), as it is a member of the Cartesian space. For rotations parameterized as quaternions, a fast on-manifold average is perfomed by finding the normalized eigenvector associated to the largest eigenvalue of a matrix built from the set of quaternions $\mathbf{q} \in \mathcal{S}^3$ to be averaged [98], as shown in (41).

$$\begin{aligned} &\mathbf{w} = [w_1, w_2, \dots, w_l], \quad \mathbf{q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_l] \\ &\mathbf{M} = \mathbf{w}\mathbf{q} = \begin{bmatrix} w_1\mathbf{q}_{1x} & w_2\mathbf{q}_{2x} & \dots & w_l\mathbf{q}_{lx} \\ w_1\mathbf{q}_{1y} & w_2\mathbf{q}_{2y} & \dots & w_l\mathbf{q}_{ly} \\ w_1\mathbf{q}_{1z} & w_2\mathbf{q}_{2z} & \dots & w_l\mathbf{q}_{lz} \\ w_1\mathbf{q}_{1w} & w_2\mathbf{q}_{2w} & \dots & w_l\mathbf{q}_{lw} \end{bmatrix} \\ &\mu_q = \text{eigvect}\left\{ \max_{eigval} \left(\mathbf{M}\,\mathbf{M}^T \right) \right\} \\ &\mu_q \in \mathcal{S}^3 \mapsto \mu_\theta \in \mathbb{R}^3 \end{aligned} \tag{41}$$

By mapping the average quaternion $\mu_q \in \mathcal{S}^3$ back to Cartesian space in the form of Euler angles $\mu_\theta \in \mathbb{R}^3$, the average transform is recovered $\bar{\mathbf{T}}\{\mu_t \in \mathbb{R}^3, \mu_\theta \in \mathbb{R}^3\} \in \mathcal{SE}(3)$. Lastly, the average covariance of the transform estimates $\bar{\mathbf{Q}} \in Sym_d^+$ is computed on-manifold as well through an iterative re-projection to the tangent space [99]:

$$\mu_{\mathbf{Q}_{v+1}} = \mu_v^{\frac{1}{2}} \exp\left(\frac{1}{l} \sum_{i=1}^{l} \log\left(\mu_v^{\frac{-1}{2}} \mathbf{Q}_i \mu_v^{\frac{-1}{2}} \right) \right) \mu_v^{\frac{1}{2}} \tag{42}$$

Where v indicates the index of the iteration. In the end, the output of the voting system is a transform $\bar{\mathbf{T}} \in \mathcal{SE}(3)$ with an associated covariance $\bar{\mathbf{Q}} \in Sym_d^+$, which enables dealing with local minimum registrations due to non-optimal hyperparameters.

4.3 Error State Kalman Filter (ESKF)

The ESKF is a filtering technique derived from the Extended Kalman Filter or *EKF* [57]. It also receives the name of *indirect Kalman Filter*, since it filters the states' error instead of the nominal states directly; in contrast to the non-linear dynamics of the states, which have to be linearized in the EKF, the linear nature of the error dynamics can be exploited to produce optimal Kalman updates.

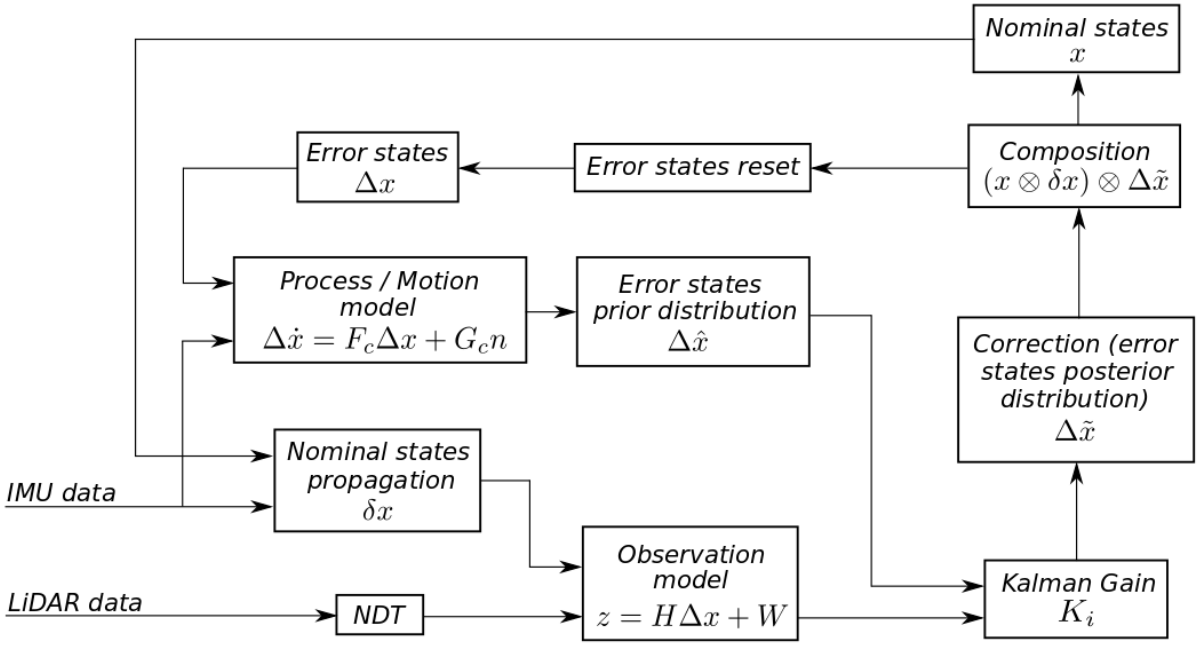

Figure 18: Block diagram representation of an Error State Kalman Filter (ESKF).

State vector The first step is to define the nominal states vector. In this dissertation, the robot's global pose (i.e. position $\mathbf{p}_W \in \mathbb{R}^3$ and orientation $\mathbf{q}_W \in \mathcal{S}^3$) and linear velocity $\mathbf{v}_W \in \mathbb{R}^3$ will be estimated, along with the intrinsic IMU parameters, the accelerometer $\mathbf{b}^a$ and gyroscope $\mathbf{b}^\omega$ biases, and the extrinsic transformation $\mathbf{T}_e \in \mathcal{SE}(3) \mapsto \{{}^L_I\tau \in \mathbb{R}^3,\ {}^L_I\phi \in \mathcal{SO}(3)\}$ between the IMU and the LiDAR.

$$\mathbf{x} = \left[\mathbf{p}_W, \mathbf{v}_W, \mathbf{q}_W, \mathbf{b}^a, \mathbf{b}^\omega, {}^L_I\tau, {}^L_I\phi\right]^T \tag{43}$$

The superscripts and subscripts ${}^L\bullet$ and ${}_I\bullet$ denote the LiDAR and IMU local frames, respectively, and the subscript $\bullet_W$ indicates that variable $\bullet$ is referred to the global reference frame. Note that the choice of notation for the orientation $\mathbf{q}_W$ anticipates that it will be parameterized through a unit quaternion $\mathbf{q}_W \in \mathcal{S}^3$ [88], which enables a more compact treatment of rotation elements.

Nominal system dynamics Let the state equation be a differentiable, non-linear function f, that maps the current state $\mathbf{x}_i$ to the predicted state $\mathbf{x}_{i+1}$. The observation model is another

non-linear function h that provides the processed sensor measurements $\mathbf{y}_i$ at state $\mathbf{x}_i$. A generic non-linear plant model is then represented by:

$$\begin{aligned} \mathbf{x}_{i+1} &= f(\mathbf{x}_i, \mathbf{v}_i) \\ \mathbf{y}_i &= h(\mathbf{x}_i, \mathbf{w}_i) \end{aligned} \tag{44}$$

Where process and sensor white Gaussian noises $\mathbf{v}_i$ and $\mathbf{w}_i$ are embedded within f and h, respectively. Dropping the reference frame subscripts for the sake of clarity, the continuous time dynamic equations for the state vector (43) are the following [58]:

$$\begin{aligned} \dot{\mathbf{p}}_i &= \mathbf{v}_i \\ \dot{\mathbf{v}}_i &= \mathbf{R}_{\{\mathbf{q}_i\}} \left(\mathbf{a}_i - \mathbf{b}_i^a - \eta_a\right) + \mathbf{g} \\ \dot{\mathbf{q}}_i &= \frac{1}{2}\mathbf{q}_i \otimes \left(\begin{bmatrix} 0 \\ \omega_i - \mathbf{b}_i^\omega - \eta_\omega \end{bmatrix} \right) \\ \dot{\mathbf{b}}_i^a &= \eta_{b^a} \\ \dot{\mathbf{b}}_i^\omega &= \eta_{b^\omega} \\ \dot{\tau}_i &= 0 \\ \dot{\phi}_i &= 0 \end{aligned} \tag{45}$$

Where $\mathbf{g} \in \mathbb{R}^3$ is the gravity vector; the subscript i denotes a certain time instant; the quaternion representing the robot's orientation can be decomposed as $\mathbf{q} = [q_w, \mathbf{q}_v]^T$ and $\mathbf{q}_v = [q_x, q_y, q_z]$, being q_w its scalar part and $\mathbf{q}_v$ the vector or imaginary part [60]; and $\mathbf{R}_{\{\mathbf{q}_i\}}$ represents the rotation matrix corresponding to the local IMU frame quaternion $\mathbf{q}_i$ with respect to the global frame.

The linear acceleration and angular velocities measured by the IMU are denoted as $\mathbf{a}_i$ and $\omega\mathbf{i}$, respectively. Also, as presented in section 2.1, the dynamics of the IMU's accelerometer and gyroscope biases are modeled as a Brownian motions [18], characterized by the integrated white noises η_{b^a} and η_{b^ω}.

Finally, the inertial data can be employed to propagate the nominal states, thus providing a predicted state distribution after a given integration interval $[t_i, t_{i+1}]$. Assuming a dead reckoning approach (i.e. without specifying a robot motion model) [100], state propagation according to IMU data is performed as shown in (46).

$$\begin{bmatrix} \hat{\mathbf{p}}_{i+1} \\ \hat{\mathbf{v}}_{i+1} \\ \hat{\mathbf{q}}_{i+1} \\ \hat{\mathbf{b}}^a_{i+1} \\ \hat{\mathbf{v}}^\omega_{i+1} \\ \hat{\tau}_{i+1} \\ \hat{\phi}_{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_i + \delta\hat{\mathbf{p}}_i \\ \mathbf{v}_i + \delta\hat{\mathbf{v}}_i \\ \mathbf{q}_i \otimes \delta\hat{\mathbf{q}}_i \\ \mathbf{b}^a_i + \delta\hat{\mathbf{b}}^a_i \\ \mathbf{b}^\omega_i + \delta\hat{\mathbf{b}}^\omega_i \\ \tau_i + \delta\hat{\tau}_i \\ \phi_i + \delta\hat{\phi}_i \end{bmatrix} = \begin{bmatrix} \mathbf{p}_i + \mathbf{v}_i\Delta t^i_{i+1} + \frac{1}{2}\left[\mathbf{R}_{\{\hat{\mathbf{q}}_i\}}(\mathbf{a}_m - \hat{\mathbf{b}}^a_i) + \mathbf{g}\right]\left(\Delta t^i_{i+1}\right)^2 \\ \mathbf{v}_i + \left[\mathbf{R}_{\{\hat{\mathbf{q}}_i\}}(\mathbf{a}_m - \hat{\mathbf{b}}^a_i) + \mathbf{g}\right]\Delta t^i_{i+1} \\ \mathbf{R}_{\{\mathbf{q}_i\}} \exp\left\{(\omega_m - \hat{\mathbf{b}}^\omega_i)\Delta t^i_{i+1}\right\} \\ \mathbf{b}^a_i \\ \mathbf{b}^\omega_i \\ \tau_i \\ \phi_i \end{bmatrix} \tag{46}$$

Error states dynamics Before presenting the dynamics of the error states, it is necessary to announce the parameterization of the rotation error $\Delta\mathbf{q} \in \mathbb{R}^3$ in Cartesian space [60]. The

rotation error, i.e. the derivative of a unit quaternion, which belongs to the Lie algebra of s^3, can be mapped to a three-dimensional vector in the Cartesian space $\theta \in \mathbb{R}^3$ and viceversa using the *wedge* and *vee* operators, as seen in section 4.1.

$$\begin{aligned} \textit{wedge} : \mathbb{R}^3 \mapsto s^3 \; ; \quad \theta \mapsto \theta^\wedge = 2\phi \\ \textit{vee} : s^3 \mapsto \mathbb{R}^3 \; ; \quad \phi \mapsto \phi^\vee = \theta = \frac{\phi}{2} \end{aligned} \tag{47}$$

Making use of the *vee* operator, the rotation error $\Delta\theta_i$ will be defined in Cartesian space as in (48). Note that, since rotations are *not* commutative, the choice of reference frame (local frame i.e. IMU, global frame i.e. world) to which the error states are referred is important [58].

$$\begin{aligned} \Delta\mathbf{q}_i = \mathbf{q}_i \otimes \hat{\mathbf{q}}_i^{-1} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\Delta\theta_i \end{bmatrix}, \quad \Delta\theta_i \in \mathbb{R}^3 \\ \text{(Global)} \; : \quad \mathbf{q}_{i+1} = \Delta\mathbf{q}_i \otimes \mathbf{q}_i \\ \text{(Local)} \; : \quad \mathbf{q}_{i+1} = \mathbf{q}_i \otimes \Delta\mathbf{q}_i \end{aligned} \tag{48}$$

Where the operator $\otimes$ denotes the quaternion multiplication. From this point onwards, the formulation presented assumes the *global frame* as the reference (for a more in-depth explanation and specific expressions to handle locally defined errors, the reader is pointed to this document [60]). Then, the full error state dynamics are presented below [58].

$$\begin{aligned} \Delta\dot{\mathbf{p}}_i &= \Delta\mathbf{v}_i \\ \Delta\dot{\mathbf{v}}_i &= -[\mathbf{R}_{\{\hat{\mathbf{q}}_i\}}\hat{\mathbf{a}}]_x \Delta\theta_i - \mathbf{R}_{\{\hat{\mathbf{q}}_i\}}\Delta\mathbf{b}_i^a - \mathbf{R}_{\{\hat{\mathbf{q}}_i\}}\eta_a \\ \Delta\dot{\theta}_i &= -\mathbf{R}_{\{\hat{\mathbf{q}}_i\}}\Delta\mathbf{b}_i^\omega - \mathbf{R}_{\{\hat{\mathbf{q}}_i\}}\eta_\omega \\ \Delta\dot{\mathbf{b}}_i^a &= \eta_{b_a} \\ \Delta\dot{\mathbf{b}}_i^\omega &= \eta_{b_\omega} \\ \Delta\dot{\tau}_i &= 0 \\ \Delta\dot{\phi}_i &= 0 \end{aligned} \tag{49}$$

Where the rotation exponential mapping (i.e. the skew-symmetric matrix form) is denoted by $[\bullet]_x$ [18], and η_a, η_ω are white Gaussian disturbances affecting the accelerometer and gyroscope measurements, respectively. The actual acceleration $\hat{a}$ and angular velocity $\hat{\omega}$ are computed by subtracting the current estimate of the biases $\hat{b}^\bullet$ from the raw measurement provided by the sensor $\bullet_m$.

$$\hat{\mathbf{a}} = \mathbf{a}_m - \hat{\mathbf{b}}^a, \qquad \hat{\omega} = \omega_m - \hat{\mathbf{b}}^\omega \tag{50}$$

Discretized process model Equation (49) can be synthesized in matrix form defining the error state vector as $\Delta\mathbf{x} = [\Delta\mathbf{p}, \Delta\mathbf{v}, \Delta\theta, \Delta\mathbf{b}^a, \Delta\mathbf{b}^\omega, \Delta\tau, \Delta\phi]^T$, and the noise vector $\mathbf{n} = [\eta_a, \eta_\omega, \eta_{b^a}, \eta_{b^\omega}]$, where $\eta_\bullet$ represents the white Gaussian noise affecting variable $\bullet$. Thus:

$$\Delta\dot{\mathbf{x}} = \mathbf{F}_c\Delta\mathbf{x} + \mathbf{G}_c\mathbf{n} \tag{51}$$

$$\mathbf{F}_c(t) = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I_3} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & [-\mathbf{R}_{\{\hat{\mathbf{q}}\}}\,\hat{\mathbf{a}}]_x & -\mathbf{R}_{\{\hat{\mathbf{q}}\}} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{R}_{\{\hat{\mathbf{q}}\}} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_{12x3} & \mathbf{0}_{12x3} & \mathbf{0}_{12x3} & \mathbf{0}_{12x3} & \mathbf{0}_{12x3} & \mathbf{0}_{12x3} & \mathbf{0}_{12x3} \end{bmatrix} \tag{52}$$

$$\mathbf{G}_c(t) = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -\mathbf{R}_{\{\hat{\mathbf{q}}\}} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -\mathbf{R}_{\{\hat{\mathbf{q}}\}} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I_3} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I_3} \\ \mathbf{0}_{6x3} & \mathbf{0}_{6x3} & \mathbf{0}_{6x3} & \mathbf{0}_{6x3} \end{bmatrix} \tag{53}$$

To achieve a discrete time implementation, matrix $\mathbf{F}_c$ can be approximated by a Taylor series expansion of order N, assuming that it is constant during the integration step (which is reasonable for small Δt), therefore reaching the following expression [101]:

$$\mathbf{F}_d(t) = \exp\left(\mathbf{F}_c(t)\Delta t\right) = \sum_{n=0}^{N<\infty} \frac{1}{n!}\mathbf{F}_c(t)^n \Delta t^n \tag{54}$$

And from the continuous time system covariance matrix $\mathbf{Q}_c = \text{diag}\left(\sigma^2_{\eta_a}, \sigma^2_{\eta_\omega}, \sigma^2_{\eta_{b^a}}, \sigma^2_{\eta_{b^\omega}}\right)$ it is possible to derive the discrete time expression of the filter's covariance matrix, $\mathbf{Q}_d$:

$$\mathbf{Q}_d = \mathbf{Q}_c \Delta t^2 \tag{55}$$

Where the terms $\sigma^2_{\eta_a}, \sigma^2_{\eta_\omega}$ represent the variance of the accelerometer and gyroscope measurements respectively, and $\sigma^2_{\eta_{b^a}}, \sigma^2_{\eta_{b^\omega}}$ model the temporal drift of the slowly-varying biases as a variance term. These magnitudes are available in the IMU's manufacturer datasheet.

Observation model The sensor model proposed is linear and involves the direct observation of some of the error states via two measurements: the transform estimation between two subsequent poses and the refinement of the extrinsic parameters. The sensor matrix $\mathbf{H}$ and the measurement vector $\mathbf{z}$ are:

$$\begin{aligned} \mathbf{z}_i &= \mathbf{H}\Delta\mathbf{x}_i + \mathbf{W}_i \\ \mathbf{z}_i &= [\Delta\hat{\mathbf{p}}_i,\ \ \Delta\hat{\theta}_i,\ \ \Delta\hat{\tau}_i,\ \ \Delta\hat{\phi}_i]^T \\ \mathbf{H} &= \begin{bmatrix} \mathbf{I_3} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I_3} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I_3} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I_3} \end{bmatrix} \end{aligned} \tag{56}$$

Where W_i is the observation noise matrix (as defined later in (60)).

On the one hand, the pose-to-pose transform estimation is based on the Normal Distributions Transform, described in section 4.2.1. The position and orientation errors ($\Delta\hat{\mathbf{p}} \in \mathbb{R}^3$ and $\Delta\hat{\theta} \in$

$\mathbb{R}^3$ in (56), respectively) are computed in the manifold, and then mapped back to Cartesian space. This is accomplished by applying the composition operator between the NDT transform estimate $\{\mathbf{t}^L \in \mathbb{R}^3,\ \theta^L \in \mathbb{R}^3\}$ and the inertial propagation of the states in (46). Note that the extrinsic rotation $R_{\{\phi_i\}}$ and the current orientation $\mathbf{R}_{\{\mathbf{q}_i\}}$ have to be applied to the LiDAR estimation to refer it with respect to the same frame as the IMU-based propagation, i.e. the global frame.

$$
\begin{aligned}
&\bullet\ \ \{\mathbf{t}^L \in \mathbb{R}^3,\ \theta^L \in \mathbb{R}^3\} \mapsto \mathbf{T}_i^L = \begin{bmatrix} \mathbf{R}_{\{\mathbf{q}_i\}}\mathbf{R}_{\{\phi_i\}}\mathbf{R}_{\{\theta^L\}} & \mathbf{R}_{\{q_i\}}\mathbf{R}_{\{\phi_i\}}t^L \\ \mathbf{0} & 1 \end{bmatrix} \in \mathcal{SE}(3) \\
&\bullet\ \ \{\delta\hat{\mathbf{p}}_i \in \mathbb{R}^3,\ \delta\hat{\theta}_i \in \mathbb{R}^3\} \mapsto \mathbf{T}_i^I = \begin{bmatrix} \mathbf{R}_{\{\delta\hat{\theta}_i\}} & \delta\hat{\mathbf{p}}_i \\ \mathbf{0} & 1 \end{bmatrix} \in \mathcal{SE}(3) \\
&\longrightarrow \quad \mathbf{\Delta T_i} = (\mathbf{T}_i^I)^{-1}\,\mathbf{T}_i^L \in \mathcal{SE}(3) \mapsto \{\Delta\hat{\mathbf{p}}_i \in \mathbb{R}^3,\ \ \Delta\hat{\theta}_i \in \mathbb{R}^3\}
\end{aligned}
\tag{57}
$$

Where $R_{\{\theta\bullet\}}$ indicates the rotation matrix $\mathbf{R} \in \mathcal{SO}(3)$ associated to the mapping of the Euler angles $\theta \in \mathbb{R}^3$ expressed in the $\bullet$ reference frame, and $\delta\hat{\theta}_i \in \mathbb{R}^3$ is the Cartesian space representation of $\delta\hat{\mathbf{q}}_i \in \mathcal{S}^3$ in (46). The covariance of this observation $\mathbf{\Sigma_p} \in \mathbb{R}^{6x6}$ will be propagated from the uncertain variables following the formulation in subsection 4.1.2.

The extrinsic parameters refinement is based on the assumption that the robot follows a rigid body motion model. Given this condition, for any robot pose there exists a chain of rotations and transformations that aligns the LiDAR frame with the initial pose, W as denoted by Figure 19. Thus, the LiDAR-to-LiDAR frame transformation matrix $\mathbf{T}_{L_{i-1}}^{L_i} \in \mathcal{SE}(3)$ is not only computed by the NDT, but can also be estimated by having precise knowledge of the extrinsic IMU-LiDAR parameters and the robot poses:

$$
\mathbf{T}_{\mathbf{L_{i-1}}}^{\mathbf{L_i}} = \left(\mathbf{T}_{\mathbf{I_{i-1}}}^{\mathbf{L_{i-1}}}\right)^{-1}\ \mathbf{T}_{\mathbf{I_{i-1}}}^{\mathbf{W}}\ \left(\mathbf{T}_{\mathbf{I_i}}^{\mathbf{W}}\right)^{-1}\ \mathbf{T}_{\mathbf{I_i}}^{\mathbf{L_i}}
\tag{58}
$$

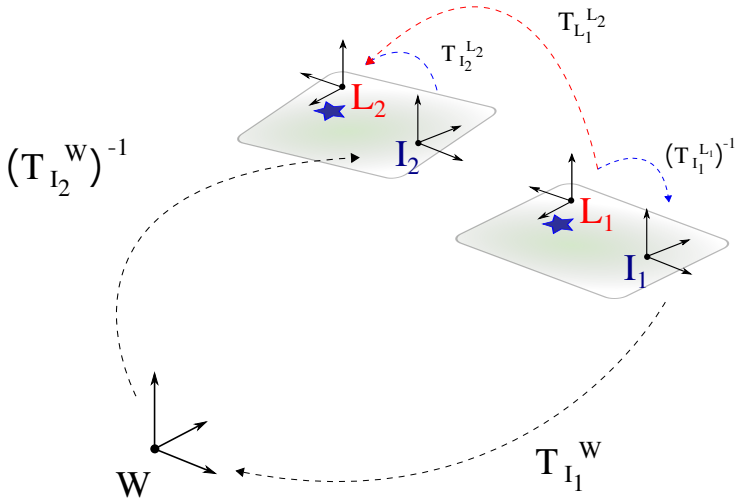

Figure 19: Two local LiDAR frames can be aligned in two ways: using the transform matrix estimated by the NDT method, or using the extrinsic parameters and the global pose relative to an initial reference.

The only unknown in this equation is the extrinsic transformation matrix for the current pose, $\mathbf{T}_{I_i}^{L_i} \in \mathcal{SE}(3) \mapsto \{\mathbf{t}_e \in \mathbb{R}^3,\ \phi_e \in \mathbb{R}^3\}$, which can be isolated from (58). Then, the observed extrinsics error is computed analogously to (59): through the composition operation in the manifold between the current extrinsic parameters $\Gamma_i \in \mathcal{SE}(3) \mapsto \{\tau_i \in \mathbb{R}^3,\ \phi_i \in \mathbb{R}^3\}$ and the observation.

$$\longrightarrow \quad \mathbf{\Delta\Gamma_i} = (\mathbf{\Gamma}_i)^{-1}\, \mathbf{T}_{I_i}^{L_i} \in \mathcal{SE}(3) \mapsto \{\Delta\hat{\tau}_i \in \mathbb{R}^3,\ \ \Delta\hat{\phi}_i \in \mathbb{R}^3\} \tag{59}$$

Again, the covariance matrix $\mathbf{\Sigma_e} \in \mathbb{R}^{6x6}$ associated with this observation is propagated from the uncertain poses in (58), as per the propagation rules discussed in section 4.1.2. Finally, the noise matrix W_i in 56 is defined accordingly:

$$\mathbf{W} = \begin{bmatrix} \mathbf{\Sigma_p^2} & \mathbf{0}_6 \\ \mathbf{0}_6 & \mathbf{\Sigma_e^2} \end{bmatrix} \tag{60}$$

Filter prediction and update The error states are modeled by a Gaussian probability distribution $\Delta x \sim \mathcal{N}(\Delta\bar{x}, P)$, characterized by a mean $\Delta\bar{x} \in \mathbb{R}^{21}$ and a covariance $P \in \mathbb{R}^{21x21}$. In the prediction step, the filter estimates the prior distribution of the error states by propagating the previous posterior distribution as follows:

$$\begin{aligned} \Delta\bar{\mathbf{x}}_{i+1|i} &= \mathbf{F}_d \Delta\bar{\mathbf{x}}_{i|i} \\ \mathbf{P}_{i+1|i} &= \mathbf{F}_d \mathbf{P}_{i|i} \mathbf{F}_d^T + \mathbf{G}_c \mathbf{Q}_d \mathbf{G}_c^T \end{aligned} \tag{61}$$

Note that this is a linear parameter-varying model, since the dynamics and noise matrices ($\mathbf{F}_d$ and $\mathbf{G}_c$, respectively) depend on the nominal states and the measured data. Then, the Kalman Gain $\mathbf{K}$ can be computed by manipulating the error states prior distribution and the observation model (sensor matrix $\mathbf{H}$ and measurement noise $\mathbf{W}$) and applied to the observed error states (i.e. residuals $\mathbf{z}_i$) to obtain a correction factor $\Delta\tilde{\mathbf{x}}_i$.

$$\mathbf{K}_i = \mathbf{P}_{i+1|i}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{i+1|i}\mathbf{H}^T + \mathbf{W}\right)^{-1} \tag{62}$$

$$\Delta\tilde{\mathbf{x}}_{i+1|i+1} = \mathbf{K}_{ii} \tag{63}$$

Finally, the nominal states are updated through said correction.

$$\begin{aligned} \mathbf{x}_{i+1} &= (\mathbf{x}_i \oplus \delta\mathbf{x}_i) \oplus \Delta\tilde{\mathbf{x}}_{i+1|i+1} \\ \mathbf{P}_{i+1|i+1} &= (\mathbf{I} - \mathbf{K}_i\mathbf{H})\, \mathbf{P}_{i+1|i}\, (\mathbf{I} - \mathbf{K}_i\mathbf{H})^T + \mathbf{K}_i \mathbf{W}_i \mathbf{K}_i^T \end{aligned} \tag{64}$$

Where $\oplus$ denotes the generic composition operator, which is the trivial summation for all elements except for the members of the rotation manifold; they are expressed in quaternion form and should follow Equation (48) for the error quaternion multiplication. The state covariance

update step is done using the Joseph form, which guarantees its symmetry and positive definiteness [60].

$$\mathbf{x}_{i+1|i+1} = \begin{bmatrix} \mathbf{p}_i + (\delta\mathbf{p}_i + \Delta\tilde{\mathbf{p}}_i) \\ \mathbf{v}_i + (\delta\mathbf{v}_i + \Delta\tilde{\mathbf{v}}_i) \\ \exp\left(\frac{1}{2}\left(\delta\theta_i + \Delta\tilde{\theta}_i\right)\right) \otimes \mathbf{q}_i \\ \mathbf{b}_i^a + (\delta\mathbf{b}_i^a + \Delta\tilde{\mathbf{b}}_i^a) \\ \mathbf{b}_i^\omega + (\delta\mathbf{b}_i^\omega + \Delta\tilde{\mathbf{b}}_i^\omega) \\ \tau_i + (\delta\tau_i + \Delta\tilde{\tau}_i) \\ \exp\left(\frac{1}{2}\left(\delta\phi_i + \Delta\tilde{\phi}_i\right)\right) \otimes \phi_i \end{bmatrix} \tag{65}$$

Error states reset Once the correction term is injected into the nominal states (64), the error states are reset through a reset function $g(\Delta\tilde{x}_{i+1|i+1})$. The mean and covariance of the Gaussian distribution is updated as per (66).

$$\Delta\mathbf{x}_{i+1} = \mathbf{0}_{21} \quad \mathbf{P}_{i+1|i+1} = \mathbf{G}\mathbf{P}_{i+1|i+1}\mathbf{G}^T \tag{66}$$

The rotation elements error term in the Jacobian of the reset function with respect to the states, $\mathbf{G}$, is neglected for simplicity's sake, leading to an identity matrix $\mathbf{G} = \mathbf{I}_{21}$ (for a more precise expression, the reader is referred to section 6.3 in [60]).

4.4 Implementation

The method presented herein has been fully implemented in C++. *Eigen* [76] library is used to deal with linear algebra and Lie algebra mappings, such as quaternion and rotation matrices from Euler angles in Cartesian space and viceversa, and operators like quaternion multiplication. The LiDAR data is stored as base *Point Cloud Library* [77] point cloud objects.

The online behavior of the filtering algorithm is embedded in a *ROS* [79] node: IMU and LiDAR data are collected on callback functions and stored chronologically on double-ended queues. Since these data packages have a timestamp associated to them, this information can be used to integrate inertial data between point cloud timestamps; note that interpolating NDT-based transform estimates is significantly harder than interpolating IMU data, which is the reasoning behind the decision of pivoting on LiDAR timestamps.

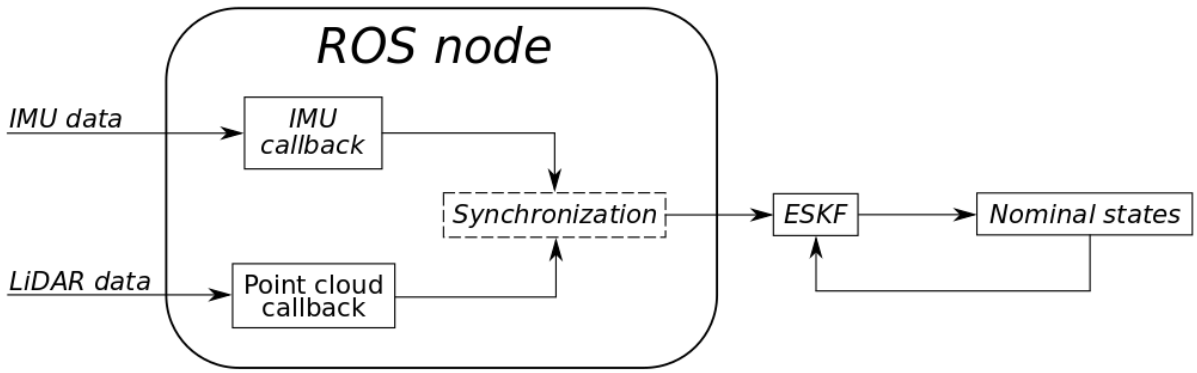

Figure 20: Block diagram of the implementation of the online filtering algorithm in a *ROS* node.

A multi-threaded implementation of the NDT algorithm is available in the *ndt_omp* library [102], which has been extended ever so slightly to provide access the Hessian matrix computed during the matching process (to be used as an estimate of the transform covariance 4.2.1). The on-manifold uncertainty propagation and the filtering algorithm are custom implementations; the latter presents some dependencies on the Bayesian filtering library *fl* [103], a general purpose library dedicated to working with common filtering algorithms.

CHAPTER 5

Experimental results

In order to assess the implementation of the calibration and state estimation algorithms, these have been tested using both synthetically generated data and data collected from real hardware. This section will review the evaluation tests devised to assess the correctness and performance of the techniques explained in this dissertation, comparing results with ground truth data (whenever possible) and benchmarking against other state of the art methods that attempt to solve the same problem.

5.1 Hardware employed

Two sources of real sensor data have been involved in the experiments. On the one hand, own-acquired LiDAR-inertial data has been collected using a custom multi-sensor rig, designed by the company *Scaled Robotics* [104] and mounted on a *Jackal* robotic platform manufactured by *Clearpath Robotics* [105]. On the other hand, publicly available dataset *KITTI* [106] designed for autonomous driving software evaluation was collected using a multi-sensor rig rigidly attached on top of a vehicle.

5.1.1 Clearpath Jackal (*Scaled Robotics*)

Figure 21: Gazebo model of the Jackal robotic platform with the multi-sensor configuration (only the ones in the scope of this work are shown).

The sensor rig features three cameras, two LiDARs and an IMU (Figure 21). However, the scope of this project is limited to the interaction between the horizontally mounted LiDAR, a *Velodyne VLP-16* [107] by *Velodyne*, and the IMU, a *Microstrain 3DM-GX5-45* [108]. Their specifications can be accessed in the corresponding citations.

Note that the IMU coordinate frame is in *NED* convention ($x \rightarrow$ North, $y \rightarrow$ East, $z \rightarrow$ Down), which is common in aerial robotics. However, since *ROS* assumes *ENU* convention ($x \rightarrow$ East, $y \rightarrow$ North, $z \rightarrow$ Up) in its 3D visualization tool, *RViz*, the IMU data will be transformed from *NED* to *ENU* reference frame before being processed for the sake of coherent visualization.

5.1.2 Volkswagen Passat (*KITTI*)

The sensor rig of the *KITTI* dataset is mounted on top of an autonomously driven vehicle. Despite the availability of multiple devices, the scope of this dissertation is limited to work with the LiDAR, a *Velodyne HDL-64E* [109] by *Velodyne*, and the IMU, an *OXTS RT-3003* [110]. The

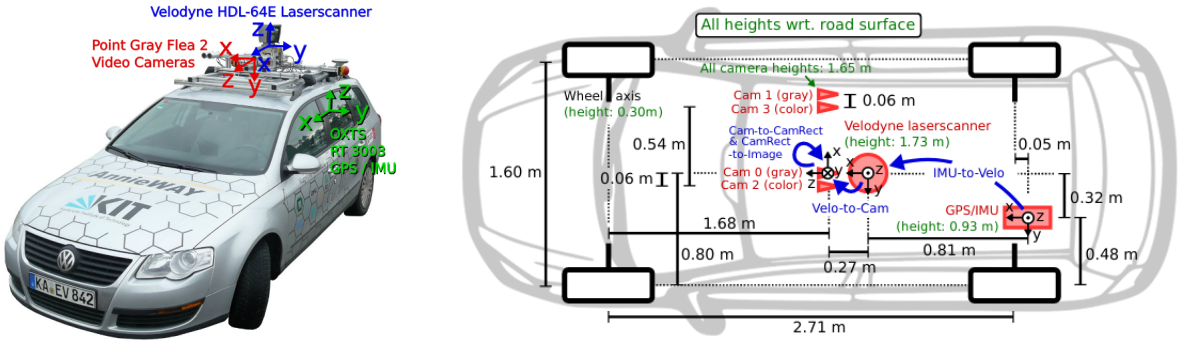

Figure 22: Vehicle and sensor rig employed for data acquisition in the *KITTI* dataset. Ground truth extrinsic parameters [106; 111].

raw data available in the *KITTI* project's site has been automatically formatted into *Rosbags* files for compatibility reasons with the *ROS* node implementation. The public repository *kitti2bag* [112] provides the necessary tools to do so.

5.2 Offline calibration

The offline calibration tests have been performed on the Jackal platform (5.1.1). To have a reasonable initial estimation of the extrinsic parameters to be optimized, the *Gazebo* [113] model of the robot is used. The IMU biases are unknown a priori, as well as the temporal shift between the IMU and the LiDAR.

Including *every* available LiDAR point (that has been successfully associated to a plane, that is) in the factor graph leads to prohibitive processing times [38]. For this reason, a maximum density of points per plane is established: only the *best* k point-to-plane associations (i.e. the k points whose computed point-to-plane distance is smaller) are included in the optimization.

Furthermore, LiDAR points that lie in the vicinity of the intersection between two planes $P_i, P_j \in \mathcal{C}$ are automatically discarded, since they are prone to be noisy associations. This is accomplished by computing the distance d_{ij} as in (25), from the observation x_i to all planes $P_j \in \mathcal{C}, j = \{1, 2, \ldots, J\}$, and comparing it to a proximity threshold χ_d; the point is discarded if more than one point-to-plane distance is below the threshold.

$$\begin{cases} D = \{\mathbf{x}_i \mid d_{ij}(P_j) < \chi_d\} \\ \text{Discard } x_i \iff |D| > 1 \end{cases} \tag{67}$$

The data rate of the IMU is estimated at around $\mathbf{250}[\mathbf{Hz}]$, while the LiDAR provides point sectors (as in 5) at around $\mathbf{753}[\mathbf{Hz}$ (i.e. the LiDAR frequency is approximately three times higher, hence the need for upsampling IMU measurements).

5.2.1 Point-to-plane association

To make sure that the LiDAR factors are included in the optimization coherently, each point in the point cloud has been *colorized* according to the plane is has been associated to (or the lack thereof). This is enabled by *PCL*'s base point cloud class, and responds purely to visualization purposes.

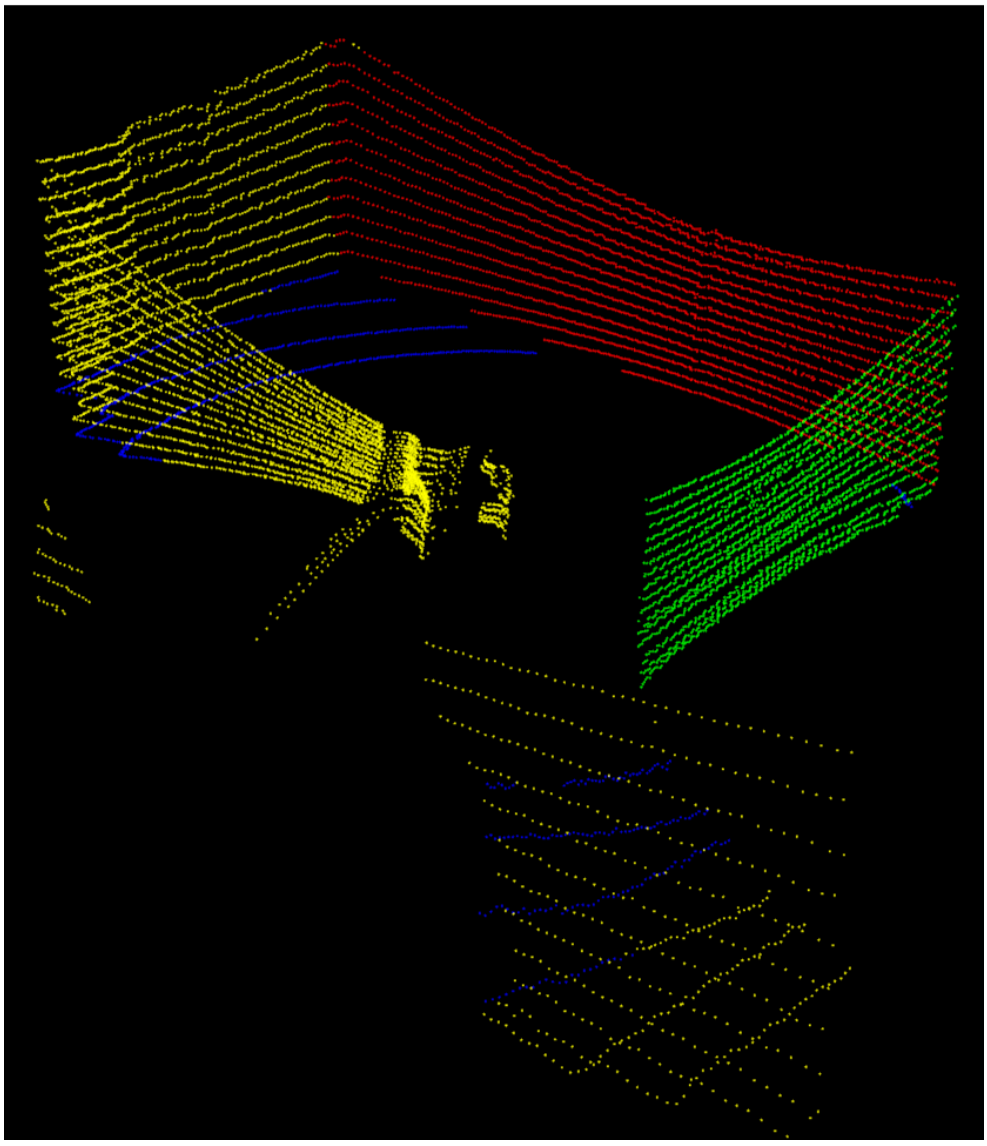

Figure 23: *PCL* point cloud with colorized points, according to point-to-plane associations.

Red, green and blue points belong to one of the planes of the calibration map; for Figure 23 in particular, blue points have been associated to the ground plane, while red and green points lie in a pair of intersecting walls. Yellow points have *not* been assigned to any plane of the calibration map. The association threshold δ_a used in this figure is $\delta_a = 0.05[m] = 5[cm]$.

Note how the orthogonality condition announced in section 3.4 is met: a corner in a room allows to find two perpendicular planes (i.e. the walls), which are in turn perpendicular to the ground. Moreover, note in Figure 24 how points around the intersection of planes (ground-wall, wall-wall) are missing, because they have been discarded as per (67) using a threshold $\chi_d = 0.05[m] = 5[cm]$ as well.

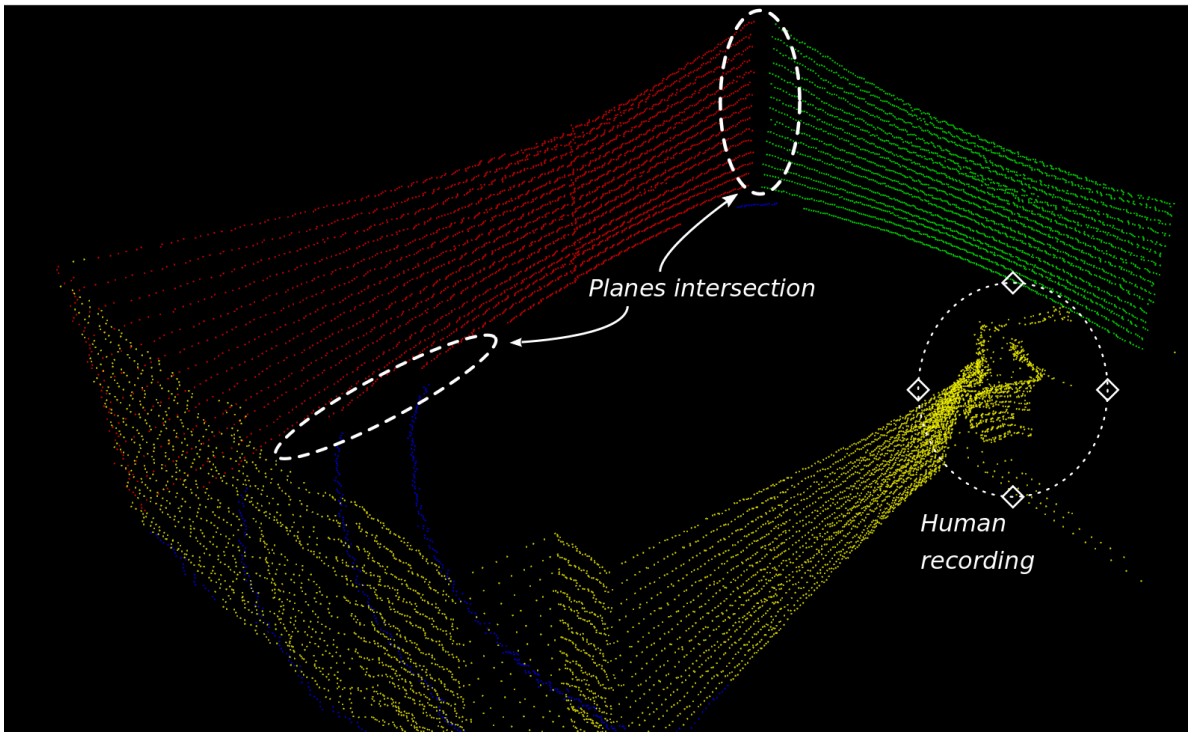

Figure 24: LiDAR points in the vicinity of the intersection between planes are discarded for the factor graph.

Finally, note that the point cloud shown in Figures 23 and 24 are not processed internally by the software as shown: the visualization has been generated after merging all LiDAR sectors that form a 360° scan, for the sake of understandability.

5.2.2 Optimization convergence tests

As discussed in section 3.5, it is necessary to stimulate the robot by performing all possible motions in the three-dimensional space, so that the generated data encodes enough information to yield a reliable estimation. Since the Jackal platform motion controller uses a differential drive configuration, it can only translate in the XY plane and rotate about the Z axis (also known as *yaw* angle). For this reason, the excitation has been generated by a human operator holding the robot in their arms (as can be interpreted from the silhouettes in Figures 23 and 24).

In order to obtain a first estimate of the optimal set of states S^* (as in (20)), the calibration problem was solved initially considering constant IMU biases $\mathbf{b}^a$, $\mathbf{b}^\omega$ and time shift δt to simplify

the problem. The temporal window size for GP training was selected at $1.0[s]$, with an overlap ratio of 20%.

Ceres optimization routine was run on a 10 $[s]$ *Rosbag* file, considering 100 keyframes with a maximum of 160 points per plane, and rejecting points in planes intersection considering a $0.1[m] = 10[cm]$ proximity threshold. Using a solver based on the *LBFGS* algorithm, a line search-based method, yielded the following results for the recording session depicted in Figures 23 and 24:

$$\begin{aligned}
\hat{\mathbf{b}}^a &= [-0.000156958, 0.000977093, 0.00121273]\ [m/s^2] \\
\hat{\mathbf{b}}^\omega &= [0.0470285, 0.0179541, 0.0401072]\ [rad/s] \\
\delta\hat{t} &= 0.020114\ [s] \\
\hat{\tau} &= [-0.417615, -0.137934, 0.309310]\ [m] \\
\hat{\phi} &= [0.183522, -0.332945, -0.365305, 0.849725]
\end{aligned} \tag{68}$$

Where ϕ is presented as a quaternion that assumes the layout $q = \{q_x, q_y, q_z, q_w\}$. Parameter convergence was reached considering a tolerance of $\|\Delta\mathbf{p}\| = 10^{-9}$ after a processing time of about 45 minutes. In order to assess the correctness of the result, the fact that the robot was static at the beginning of the recording enables estimating the IMU biases via inspection of the samples interpolated from the Gaussian Process model, which intrinsically filter out high frequency noise.

In other words, since the robot was not in motion, the values computed from interpolating the model correspond to the accelerometer and gyroscope biases, which read the following:

$$\begin{aligned}
b_z^a &= g - a_z = 9.80655 - a_z\ [m/s^2] \\
\mathbf{b}^a &= [0.00329621, 0.054636, 0.00032]\ [m/s^2] \\
\mathbf{b}^\omega &= [-0.00324833, -1.29782e-05, 0.000476821]\ [rad/s] \\
\tau &= [-0.139, -0.001, -0.328]\ [m] \\
\phi &= [-0.002, 0.003, -0.0007, 1.000]
\end{aligned} \tag{69}$$

Note that gravity has to be subtracted from the $\mathbf{a}_z$ estimate to obtain the remaining bias. Added to the significant deviation from the initial extrinsic parameters estimate, this suggests that the large dimensionality of the problem caused the solver to fall into a local minimum.

Upon reaching this conclusion, the parameters in (69) were used to initialize a more complex form of the optimization problem (the temporal shift parameter was set to $2[ms]$ from manual inspection of the *Rosbag* file and the data timestamps). As discussed in section 3.3, the IMU biases and temporal shift will be included within the state associated to the keyframes $\mathbf{s}_m = \{\mathbf{p}_m, \mathbf{v}_m, \mathbf{q}_m, \mathbf{b}_m^a, \mathbf{b}^\omega, \delta t_m\}$.

In this occasion, the optimization routine was run considering the same hyperparameters as in the previous experiment (in terms of training intervals and keyframe selection), but using a non-linear iterative Schur complement-based solver. The results, obtained after approximately 2 hours of processing time, are depicted in the next figures.

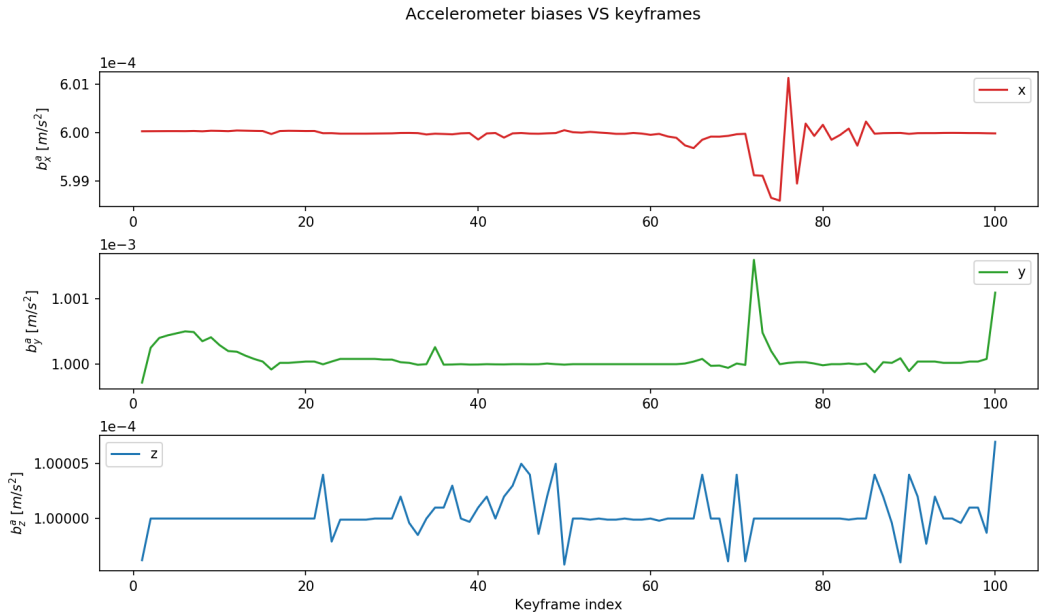

Figure 25: Optimal accelerometer bias components (x, y, z) per keyframe.

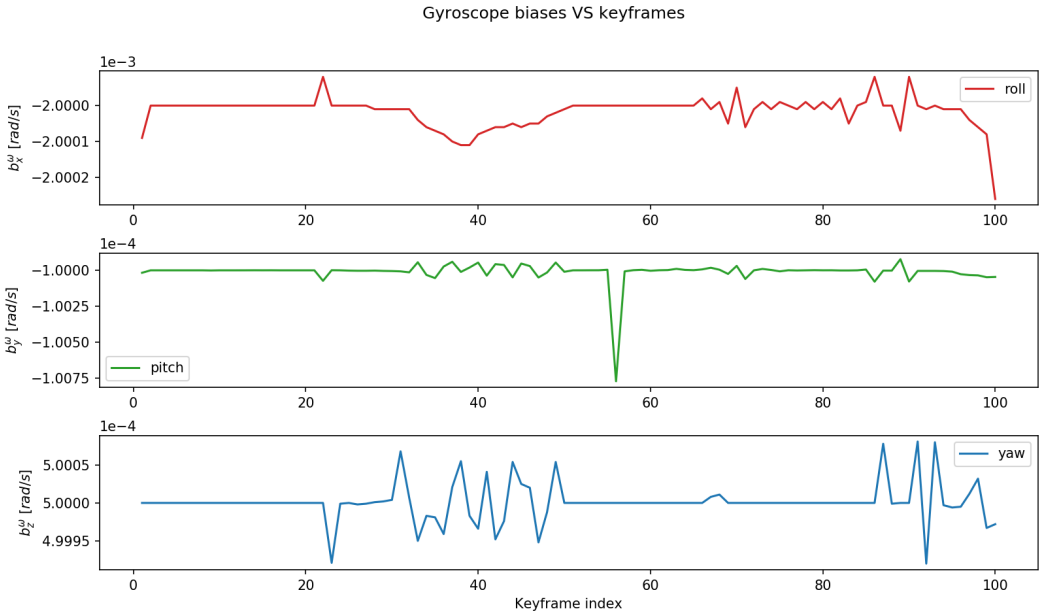

Figure 26: Optimal gyroscope bias components $(x$ (roll), y (pitch), z (yaw) per keyframe.

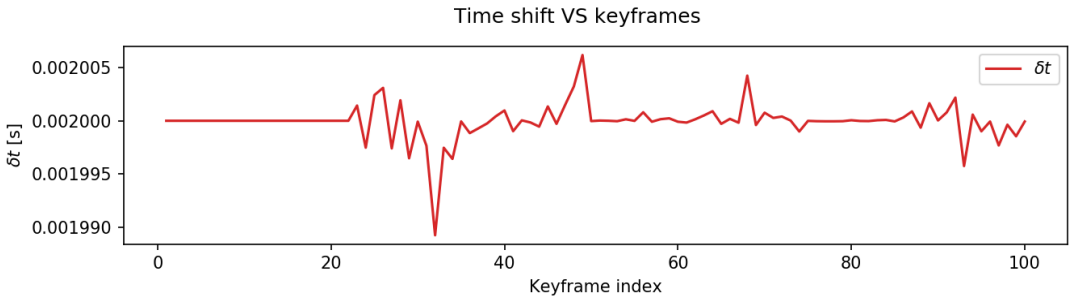

Figure 27: Optimal time shift computed per keyframe.

$$\begin{aligned} \hat{\tau} &= [-0.139, -0.001, -0.328]\ [m] \\ \hat{\phi} &= [-0.002, 0.003, -0.0007, 1.000] \end{aligned} \tag{70}$$

The plots denote consistent mean values throughout the keyframes for all parameters, with very small variations around them. The optimal extrinsic parameters converge at the initial estimation, as in (69). This behavior is sensible for well-tuned parameters, assuming that the initial estimates are already accurate. Additionally, the solver successfully converges, considering a parameter tolerance of criterion of $\|\Delta\mathbf{p}\| = 10^{-9}$.

Taking the previous considerations into account, the conclusion reached is the following: given the lack of ground truth data for IMU biases and time shift parameters, it is not possible to assess if this estimation is close to the real values. However, there is no clear proof to assume otherwise. In any case, more tests should be performed in order to ensure that the IMU excitation is enough to obtain a quality estimation.

5.3 Online state estimation

The two core modules of the state estimation block, the Normal Distributions Transform for scan matching and the Error State Kalman Filter as the filtering algorithm, will be analyzed (separately first, then in tandem for LiDAR-inertial odometry).

Note that in order to evaluate the performance of both systems ground truth data is needed. Since it has not been possible to obtain such data with the Jackal platform (more details in section 8), the upcoming tests will be based on sequences from the *KITTI* dataset.

As briefed in subsection 5.1.2, it features point-cloud, visual and inertial data recorded using an autonomous car. One should bear in mind that the magnitudes of the states and measurements in an autonomous vehicle (mainly the velocity and inertial data) are *not* directly representative of Jackal's specifications, which works at a smaller scale.

A similar remark can be made about the LiDAR model: while the one used in *KITTI* is the *Velodyne HDL-64E*, which features an array of 64 lasers, the *Velodyne VLP-16* model mounted on Jackal's sensor rig is a smaller version featuring 16 lasers. In other words, the vertical field of view of the point clouds in *KITTI* is higher and the point density is larger.

Before delving into the results, some error metrics will be defined for evaluation purposes. More of them can be found in the *OpenVINS* evaluation package, implementation included [114].

- **Root Mean Squared Error (RMSE):** the squared root of the unsigned distance between two $\mathbb{R}^n$ dimensional vectors $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$, i.e. the Euclidean norm of the difference between the two. This metric will be used to quantify errors in translation $\mathbf{t}_\epsilon \in \mathbb{R}^3$ and rotation $\theta_\epsilon \in \mathbb{R}^3$ estimations at each timestep t_i.

$$RMSE_i = \sqrt{\|\mathbf{x}_i \boxminus \hat{\mathbf{x}}_i\|_2^2} \tag{71}$$

- **Relative Pose Error (RPE):** computation of the pose error $\tilde{\mathbf{x}}_r \in \mathcal{SE}(3)$ over segments of pre-defined lengths $\mathcal{D} = [d_1, d_2, \ldots, d_J]$. This metric allows to easily compare how different localization techniques drift as the trajectory evolves.

$$\begin{aligned} \tilde{\mathbf{x}}_r &= \mathbf{x}_k \boxminus \mathbf{x}_{k+d_j} \\ RPE_{d_j} &= \frac{1}{D}\sum_{k=1}^{D}\left\|\tilde{\mathbf{x}}_r \boxminus \hat{\tilde{\mathbf{x}}}_r\right\|_2^2 \end{aligned} \tag{72}$$

- **Relative Orientation Error (ROE):** analogously to the relative pose error metric, it captures the average orientation error $\tilde{\theta} \in \mathbb{R}^3$.

$$\begin{aligned} \tilde{\theta}_r &= \theta_k \boxminus \theta_{k+d_j} \\ ROE_{d_j} &= \frac{1}{D}\sum_{k=1}^{D}\left\|\tilde{\theta}_r \boxminus \hat{\tilde{\theta}}_r\right\|_2^2 \end{aligned} \tag{73}$$

5.3.1 NDT robustness assessment

NDT versus ground truth In order to evaluate the accuracy of the NDT as a scan matching technique, the error between the rigid transformation $\tau_{ij} \in \mathbb{R}^6 \mapsto \mathbf{T}_{ij} \in \mathcal{SE}(3)$ and the ground truth transform $\mathcal{T}_{ij} \in \mathcal{SE}(3)$ between subsequent scans $[Y_i, Y_j]$ will be used as a metric. Note that the point clouds are processed as 360° scans, in contrast to the LiDAR setup adopted in calibration.

Since the ground truth transforms $\mathcal{T}_{ij} \in \mathcal{SE}(3)$ are expressed with respect to the global frame, while the NDT estimates $\tau_{ij} \in \mathbb{R}^6 \mapsto \mathbf{T}^L ij \in \mathcal{SE}(3)$ are referred to the local LiDAR frame, the latter will be projected to the world frame $\mathbf{T}_{ij}^W \in \mathcal{SE}(3)$. Note that the ground truth extrinsic transform $\mathbf{T}_e \in \mathcal{SE}(3)$ between the LiDAR and vehicle base frame is known, as well as the orientation of the car with respect to the world $\mathbf{R}_j^W \in \mathcal{SO}(3)$, and thus no error will be induced.

$$\begin{aligned} &\mathbf{T}_R = \begin{bmatrix} \mathbf{R}_j^W & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \in \mathcal{SE}(3), \quad \mathbf{T}_e = \begin{bmatrix} \mathbf{R}_e & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \\ &\mathbf{T}_{ij}^W = \mathbf{T}_R \mathbf{T}_e \mathbf{T}_{ij}^L \\ &\epsilon_{ij} = \left(\mathbf{T}_{ij}^W\right)^{-1} \mathcal{T}_{ij} \in \mathcal{SE}(3) \mapsto \{\mathbf{t}_\epsilon \in \mathbb{R}^3, \theta_\epsilon \in \mathbb{R}^3\} \end{aligned} \tag{74}$$

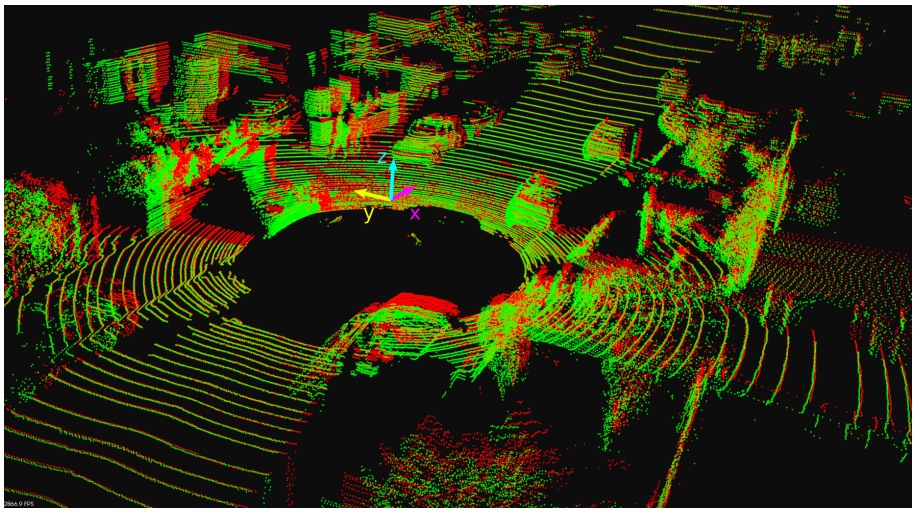

Figure 28: Two points clouds from subsequent scans, represented by red and green point clouds, from *2011_06_29_0035* sequence in *KITTI*. The highlighted reference frame at the origin is the LiDAR's.

Note that the extrinsics term $\mathbf{T}_e$ does not include the translation term, as this is a motion estimate rather than an absolute pose measurement w.r.t. the global frame (i.e. the LiDAR frame should solely be oriented towards the global frame in order to express the rigid motion coherently). The position $\mathbf{t}_\epsilon \in \mathbb{R}^3$ and orientation $\theta_\epsilon \in \mathbb{R}^3$ errors are computed for the full sequence of scans. As indicated in the last expression of (74), the rotational error will be presented in Euler angles format.

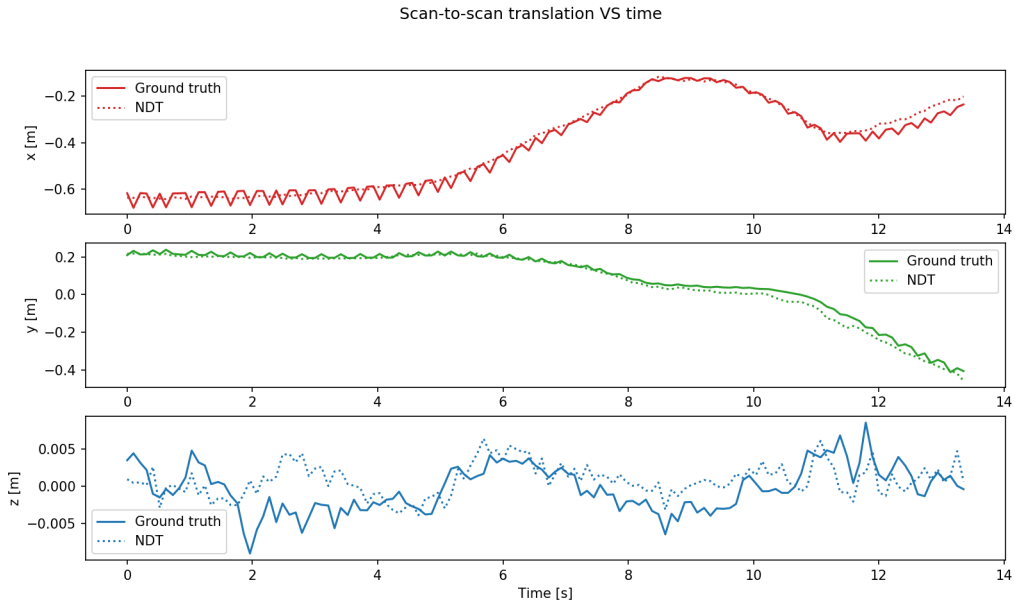

Figure 29: *KITTI* sequence: *2011_09_26_0035*. X, Y and Z ground truth (solid) and estimated (dashed) translations [m] against time [s] relative to the initial instant.

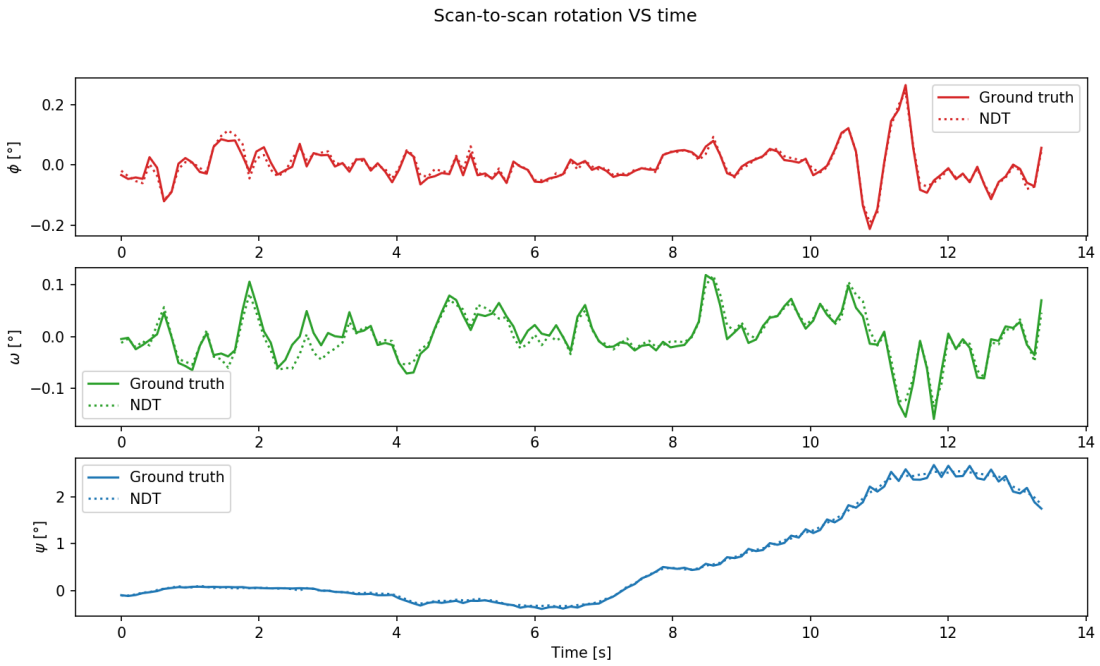

Figure 30: *KITTI* sequence: *2011_09_26_0035*. Roll (ϕ), pitch (ω) and yaw (ψ) ground truth (solid) and estimated (dashed) angles [°] against time [s] relative to the initial instant.

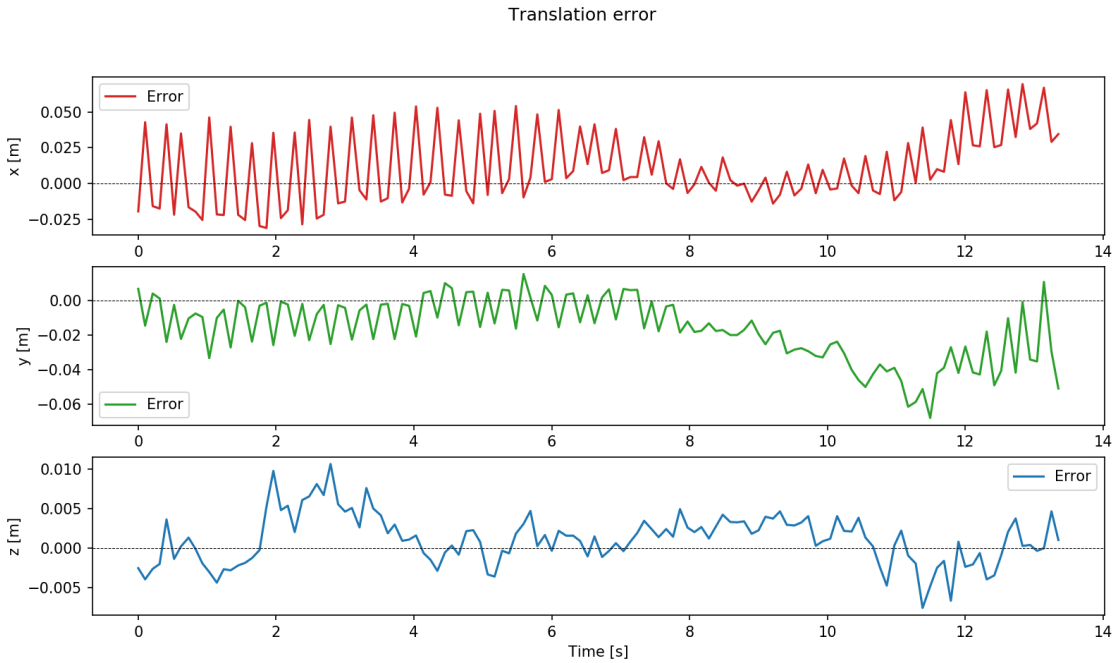

Figure 31: *KITTI* sequence: *2011_09_26_0035*. Errors in X, Y and Z translation estimates [m] against time [s] relative to the initial instant.

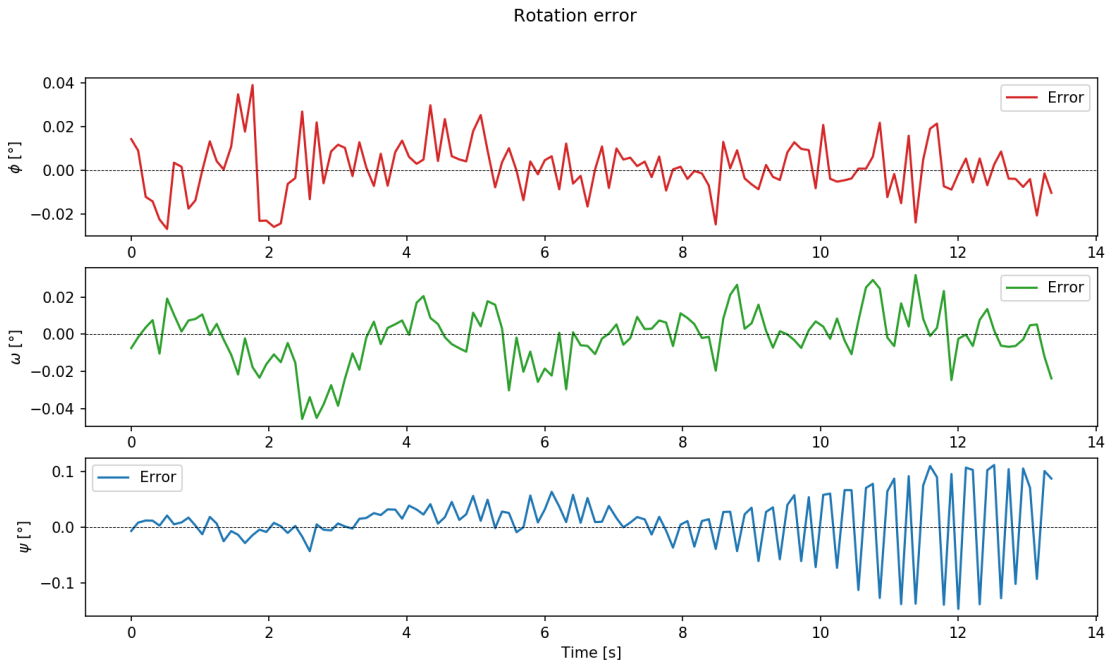

Figure 32: *KITTI* sequence: *2011_09_26_0035*. Error in roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates [°] against time [s] relative to the initial instant.

Figures 29 and 30 represent the rigid motions between subsequent scans. Thus, each value plotted against time represents a translation or a rotation; at first glance, the NDT estimates are quite close to the ground truth.

When inspecting the errors represented in Figures 31 and 32 one would then expect to find that their mean value are around zero. With the exception of some intervals (e.g. translation error in Y from second 8 onwards, translation in Z and pitch error between seconds 2 and 4), the error stabilizes around the zero.

However, the impact of these accumulated errors is significant when working with a filtering algorithm, considering the fact that the NDT observation in the global frame depends on the estimated nominal orientation (note that the ground truth value was used for this evaluation). There will be a more in-depth discussion in the filter experiments in section 5.3.2.

As discussed in section 4.2.1, the performance of the Normal Distributions Transform is affected by hyperparameters present in its algorithm. For future reference, all results presented in this experiment considered a voxel size of $\mathbf{1}[\mathbf{m}]$ and an expected outlier ratio of $\mathbf{25}\%$; the convergence criteria was set to a step size $\Delta \boldsymbol{p} = \mathbf{10^{-6}}$ and a maximum of $\mathbf{60}$ iterations.

NDT consistency If a user were to match a point cloud to an exact copy of itself, they would expect the rigid motion estimate $T_{ii} \in \mathcal{SE}(3)$ to be null, i.e. the identity rigid transformation matrix $T_{ii} = I_4$. Albeit obvious at first sight, the Newton's-algorithm based solver that the NDT implements may converge to other local minimums.

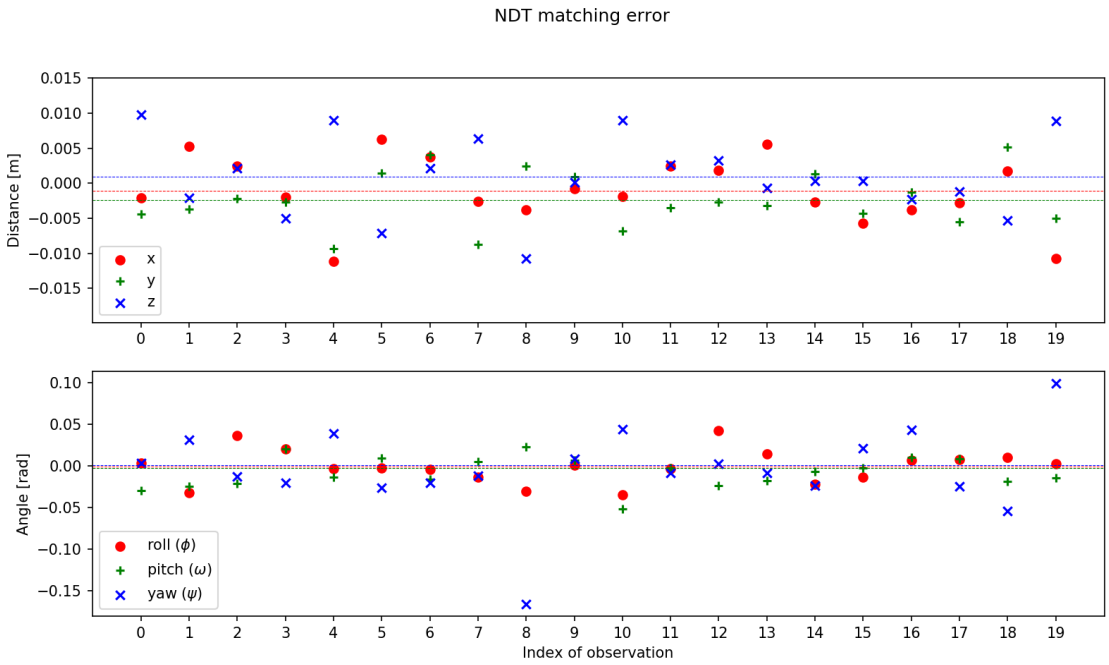

Figure 33: Scatter plot of the translation and rotation estimates obtained when matching arbitrary clouds to themselves. The mean value of each parameter is drawn as a horizontal, dashed line of the corresponding color.

To this end, a randomly selected set of clouds from one of the *KITTI* sequences was used; for each of them an NDT alignment was computed, where the same cloud was set as the source and as the target. Figure 33 depicts the individual observations and the average translation and rotation for all sampled estimations.

The expectation is confirmed by Figure 33: the translation elements and the Euler angles average is within a certain tolerance of zero $\{t = 0_{3x1} \in \mathbb{R}^3,\ \theta = 0_{3x1} \in \mathbb{R}^3\} \mapsto T = I_4 \in \mathcal{SE}(3)$. Note, however, that the sample variance is significant (specially for the translation component of the transformation, with errors in the range of centimeters), which speaks to the accuracy that can be expected from the NDT.

The hyperparameters considered in this experiment include a voxel size of $\mathbf{1}[\mathbf{m}]$ and an expected outlier ratio of $\mathbf{30}\%$. The convergence criteria was set to a step size $\Delta \boldsymbol{p} = \mathbf{10^{-6}}$ and a maximum of $\mathbf{30}$ iterations.

Hyperparameter dependence The solution that the scan registration technique converges towards is conditioned to the set of hyperparameters used. As discussed by the author of the 3D-NDT, they should be tuned according to the specifications of the sensor capturing the point clouds as well as to the size of the significant features in the scene [87]. Therefore, one can make an *a priori* estimation of what values work best.

Figure 34 shows the estimated rigid motions errors for voxel sizes of $\mathbf{1.0}[\mathbf{m}]$ and $\mathbf{2.0}[\mathbf{m}]$, and expected outlier ratios of $\mathbf{25}\%$ and $\mathbf{60}\%$. Finally, the output of the voting system (i.e. the weighted averaging of transforms $\mathbf{T}_i \in \mathcal{SE}(3)$ is also presented.

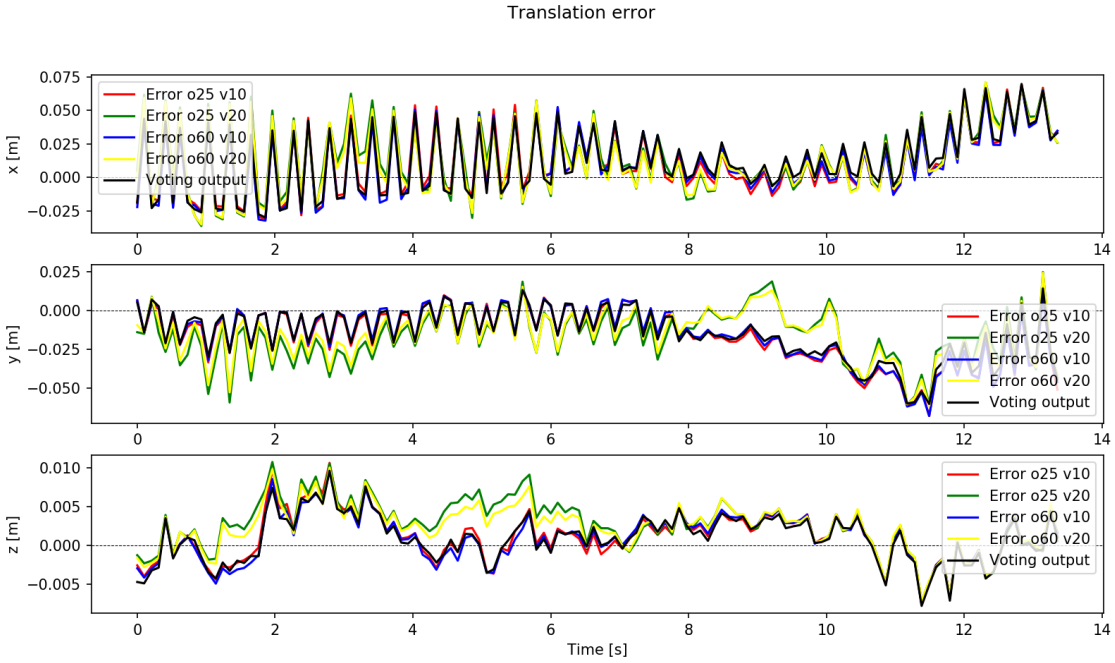

Figure 34: *KITTI* dynamic sequence: *2011_09_26_0005*. Errors in X, Y and Z translation estimates [m] against time [s] relative to the initial instant. *o25* $\mapsto$ Outlier ratio $= 25\%$, *o60* $\mapsto$ Outlier ratio $= 60\%$, *v10* $\mapsto$ Voxel size $= 2.0[m]$, *v10* $\mapsto$ Voxel size $= 2.0[m]$.

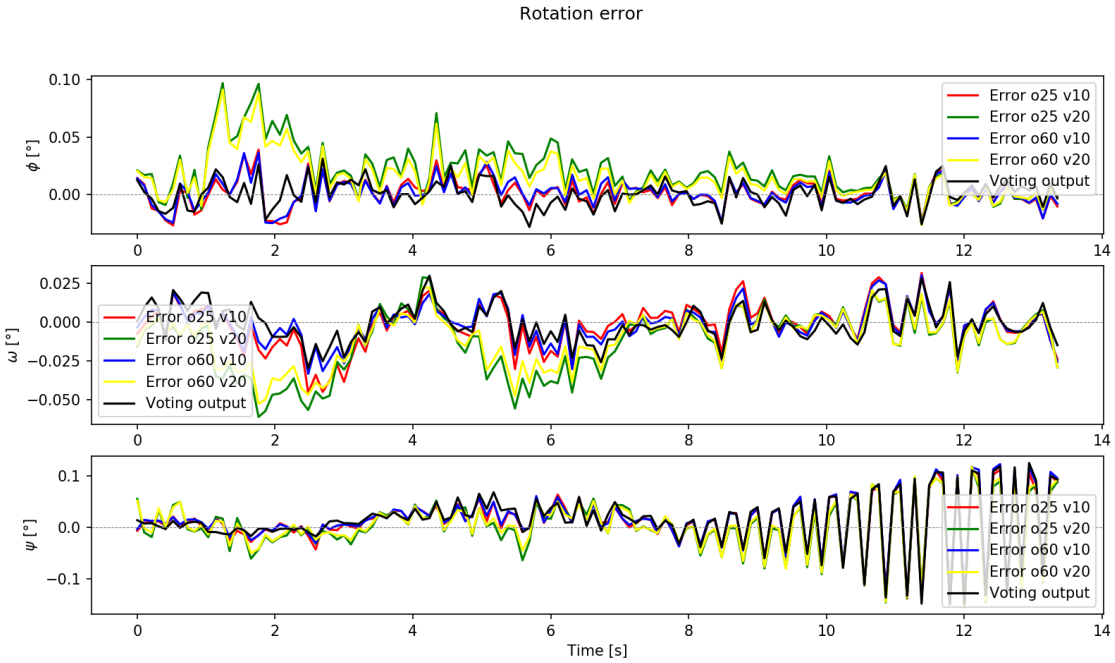

Figure 35: *KITTI* dynamic sequence: *2011_09_26_0005*. Error in roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates [°] against time [s] relative to the initial instant. *o25* $\mapsto$ Outlier ratio $= 25\%$, *o60* $\mapsto$ Outlier ratio $= 60\%$, *v10* $\mapsto$ Voxel size $= 2.0[m]$, *v10* $\mapsto$ Voxel size $= 2.0[m]$.

From a quick glance at 34 and 35 it is possible to see that the hyperparameter with the strongest influence in the estimation is the voxel size. Comparatively speaking, the ratio of expected outliers not drastically impact the rigid motion estimation. The only significant difference is the covariance estimate associated to the observed transform, which is higher the smaller the expected ratio of outliers is.

Note how the weighted average tends towards the less uncertain transforms; the averaging is not able to simply negate the estimation error, but provides a consensus for circumstances in which the variability of the NDT alignment is high.

In order to predict the rigid motion that aligns the two clouds frames it is not mandatory to estimate the covariance matrix *per se*. However, its representation plays the key role of quantifying the uncertainty of the alignment; the scale of said matrix should thus be coherent with the physical units of the associated states. This will be further discussed in experiment 5.3.2.

NDT in dynamic scenes The presence of dynamic objects in the scene poses a challenge to point cloud registration techniques. Since the rigid motion estimation is computed *relative* to the LiDAR, if the world (or some elements in it) also moves between scans the alignment $\{\mathbf{t} \in \mathbb{R}^3,\ \theta \in \mathbb{R}^3 \mapsto T \in \mathcal{SE}(3)\}$ can converge to an erroneous estimate.

Feature correspondence-based methods are specially weak to these circumstances, since even if the data association is correct, the rigid motion indicated by correspondences on dynamic objects is not representative of the LiDAR's absolute motion. For this reason, literature on scan matching dedicates some effort to detecting and removing dynamic elements before performing the point cloud alignment itself [100].

Figure 36: First two frames from *KITTI* sequence *2011_09_26_0005*, which contains dynamic elements. Camera data is presented rather than LiDAR point cloud, for the sake of understandability.

The fact that the Normal Distributions Transform does not rely on explicit point-to-point or feature-to-feature correspondences confers it with natural robustness in dynamic scenarios. Under the assumption that a comparatively large portion of the scan is actually static (e.g. road, buildings, vegetation, ...), the influence that smaller dynamic objects might have in computing the optimal rigid transform is relatively minor.

In order to test this property, a sequence from the *KITTI* dataset featuring dynamic objects in the environment such as vehicles, cyclists or pedestrians is chosen. The hyperparameters considered in this experiment include a voxel size of $\mathbf{1}[\mathbf{m}]$ and an expected outlier ratio of $\mathbf{25}\%$, which have proved to work well in other instances of the dataset. The convergence criteria was set to a step size $\Delta \boldsymbol{p} = \mathbf{10^{-6}}$ and a maximum of $\mathbf{60}$ iterations.

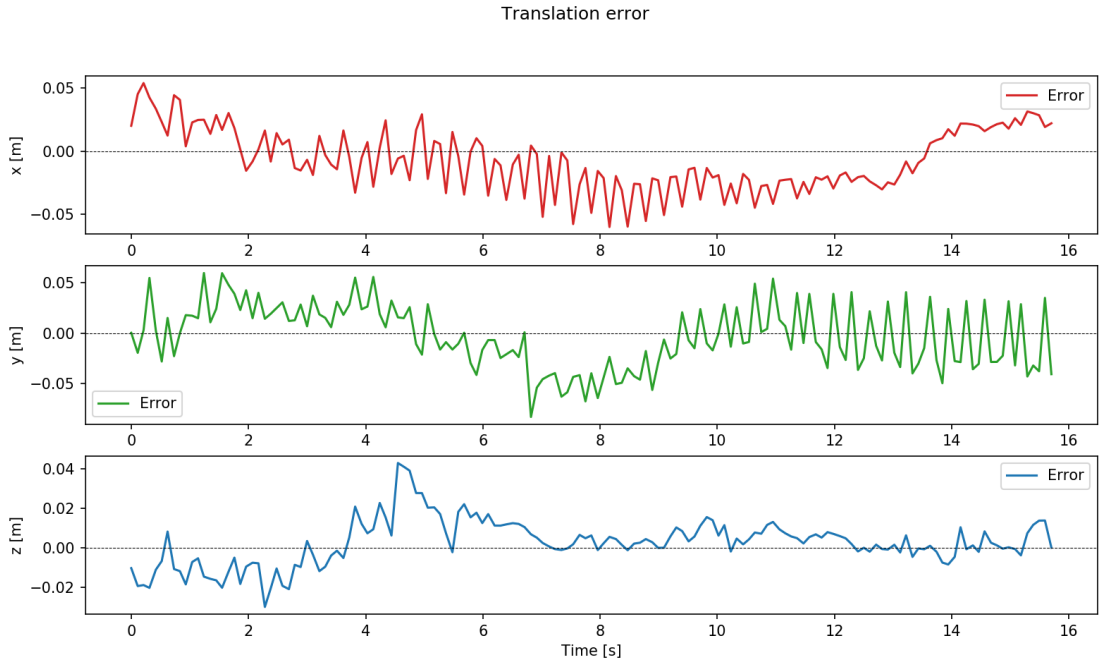

Figure 37: *KITTI* dynamic sequence: *2011_09_26_0005*. X, Y and Z ground truth (solid) and estimated (dashed) translations [m] against time [s] relative to the initial instant.

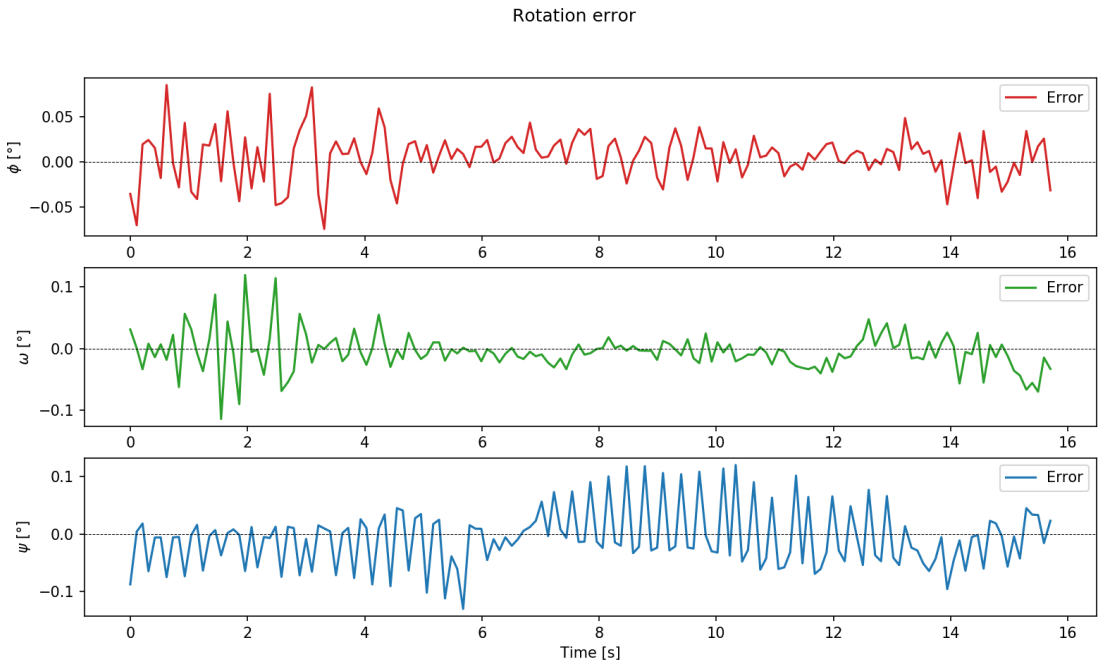

Figure 38: *KITTI* dynamic sequence: *2011_09_26_0005*. Roll (ϕ), pitch (ω) and yaw (ψ) ground truth (solid) and estimated (dashed) angles [°] against time [s] relative to the initial instant.

Figures 37 and 38 suggest that the NDT is indeed able to deal with moderate amounts of dynamic elements in the scene, as the tendency of the error is to stabilize around zero. Still, there are some intervals in which its performance deteriorates, e.g. the translation estimate in Y in the $[6.5, 8.5]$ $[s]$ interval.

Lastly, note that the results presented correspond to a particular choice of discretization, i.e. the voxel size of $\mathbf{1}[\mathbf{m}]$, albeit the same experiment could still fail for a different selection of hyperparameters.

5.3.2 Filter convergence using ground truth

To assert the correct behavior of the filtering algorithm, the Error State Kalman Filter (or simply *ESKF*), ground truth data from *KITTI* sequences is used. Since the states filtered are the error states, an observation is nothing else than the difference between the propagation of the current nominal states through inertial data and the scan matching-based odometry.

By replacing the laser odometry with the ground truth rigid motion transformations the error states observation would be computed exactly; thus, the output of the filter would be expected to converge towards the ground truth nominal states. Figures 39, 40 and 41 present the results of the test.

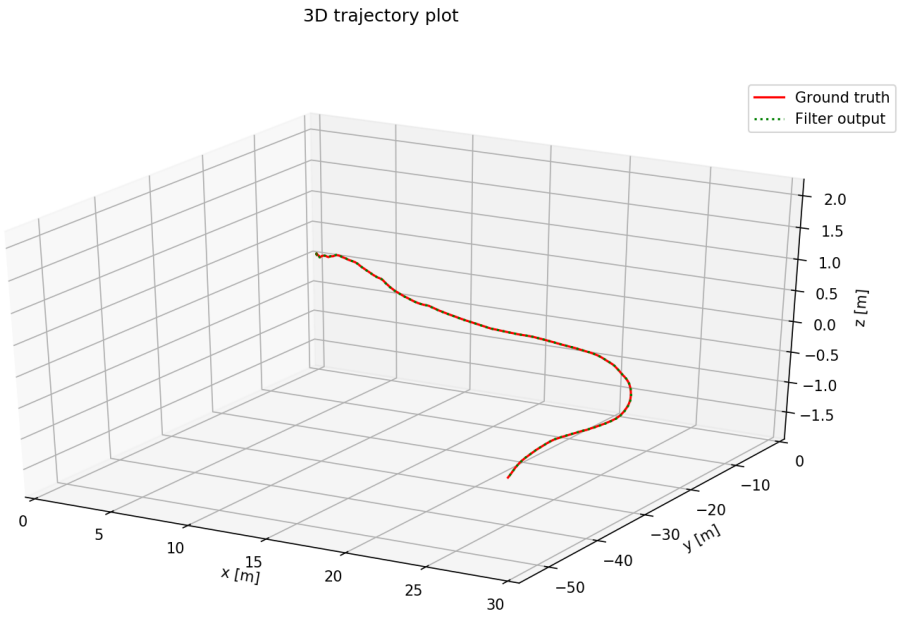

Figure 39: *KITTI* dynamic sequence: *2011_09_26_0005*. 3D trajectory described by ground truth data (solid red) and the output of the filter (dashed green), using ground truth data.

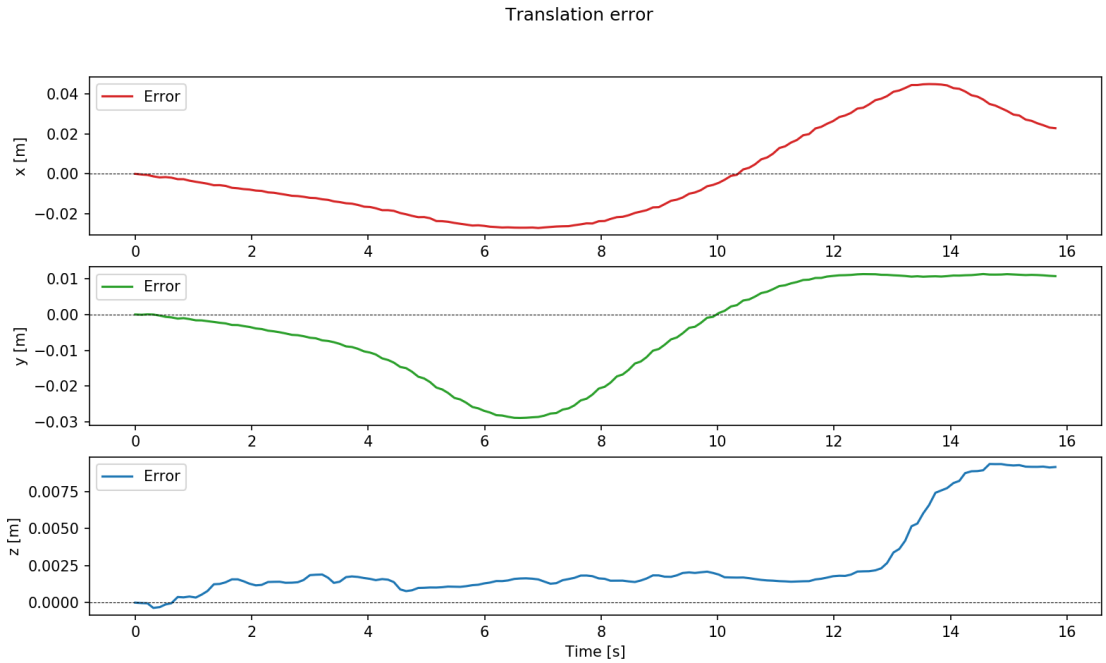

Figure 40: *KITTI* dynamic sequence: *2011_09_26_0005*. Errors in X, Y and Z translation estimates [m] against time [s] relative to the initial instant, using ground truth data.

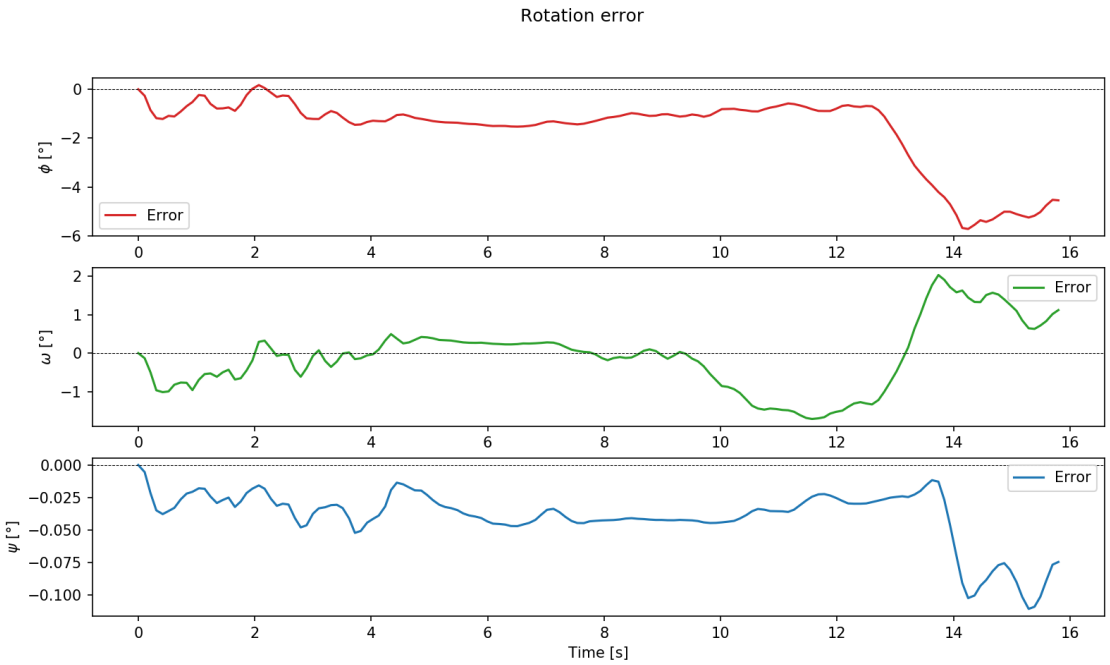

Figure 41: *KITTI* dynamic sequence: *2011_09_26_0005*. Error in roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates [°] against time [s] relative to the initial instant, using ground truth data.

As shown in Figure 39, the 3D position does indeed converge towards the ground truth nominal states. However, inspecting the error in the attitude estimation (Figure 41) raises a different point: when the inertial propagation differs from the ground truth not only a correction term is applied, but the IMU biases are updated as well. This causes a lingering inertia in the filter output, accentuated for sudden bias updates (which are caused by large estimation errors).

On the other hand, the lower the covariance of the estimation is, the higher the Kalman gain that the filter computes. In other words, the uncertainty of the observation determines the rate of convergence of the updates towards the ground truth: the smaller the scale of the covariance, the faster the convergence is and viceversa. Figures 42 and 43 prove this point.

While the green output depicted in Figures 42 and 43 considered a covariance with terms in the order of 10^{-6} in the diagonal (as in Figures 40 and 41, the red output represents the filter behavior if it were to consider the same ground truth observations, albeit using the covariance estimate of the LiDAR-based odometry (whose diagonal elements lie in the order of 10^{-1}). Indeed, the larger scale of the rotation makes the convergence rate to be too slow, causing the filter to diverge from the ground truth.

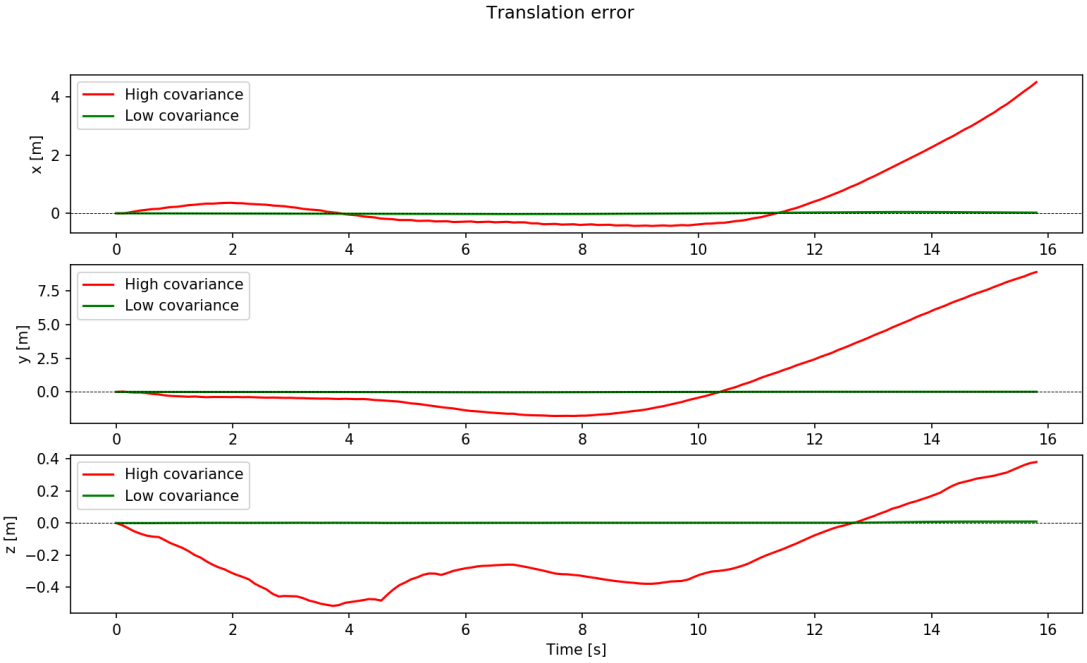

Figure 42: *KITTI* dynamic sequence: *2011_09_26_0005*. Errors in X, Y and Z translation estimates [m] against time [s] relative to the initial instant, assuming low observation covariance (red) and high observation covariance (green).

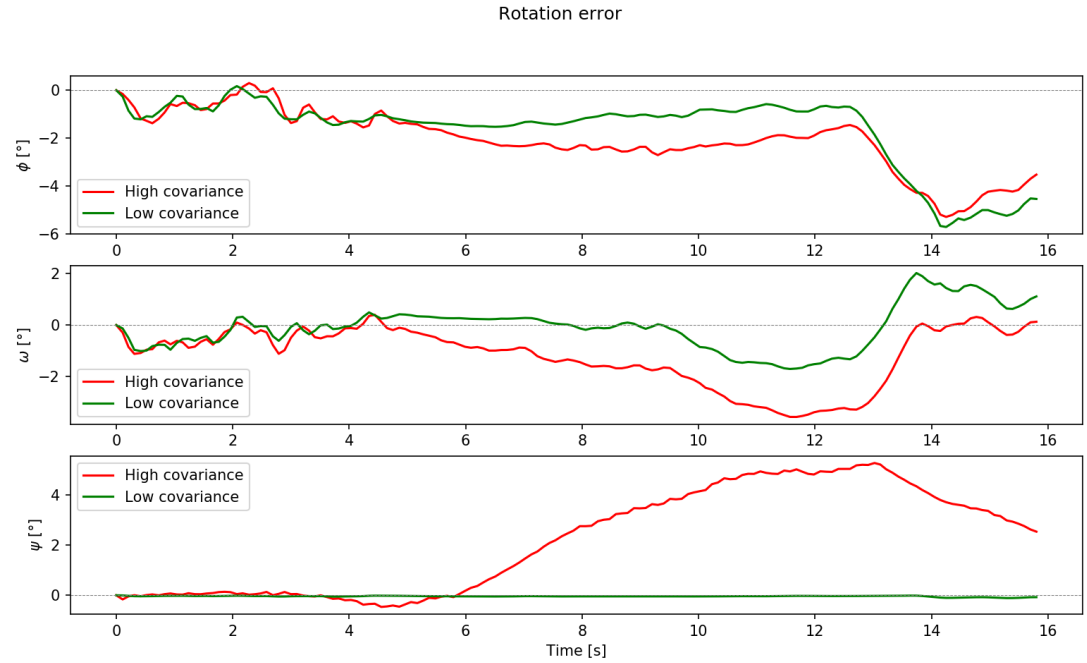

Figure 43: *KITTI* dynamic sequence: *2011_09_26_0005*. Error in roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates [°] against time [s] relative to the initial instant, assuming low observation covariance (red) and high observation covariance (green).

5.3.3 Inertial-only state estimation

The most basic form of performing localization is to only consider inertial measurements. This is prone to numerous negative effects: drift caused by the continuous integration of error, sensitivity to high frequency noise, need for precise characterization of intrinsic biases, ...

Figures 44, 45 and 46 depict the drifting effect very clearly. Another factor that has to be considered is the initialization of the nominal states: since this filter is purely based on state propagation, an inaccurate initialization would lead to large accumulations of error.

Note that, since only the IMU measurements are available, it is not possible to correct its biases online (they are not observable). From a theoretical point of view, in the implementation of the ESKF this translates to error state observations being always *zero*, as the state propagation is also computed from inertial data. Since the biases do not have any dynamics associated to the states themselves (i.e. the associated rows of the dynamic matrix of the error states (52) are zeroes), the filter is not able to update their value.

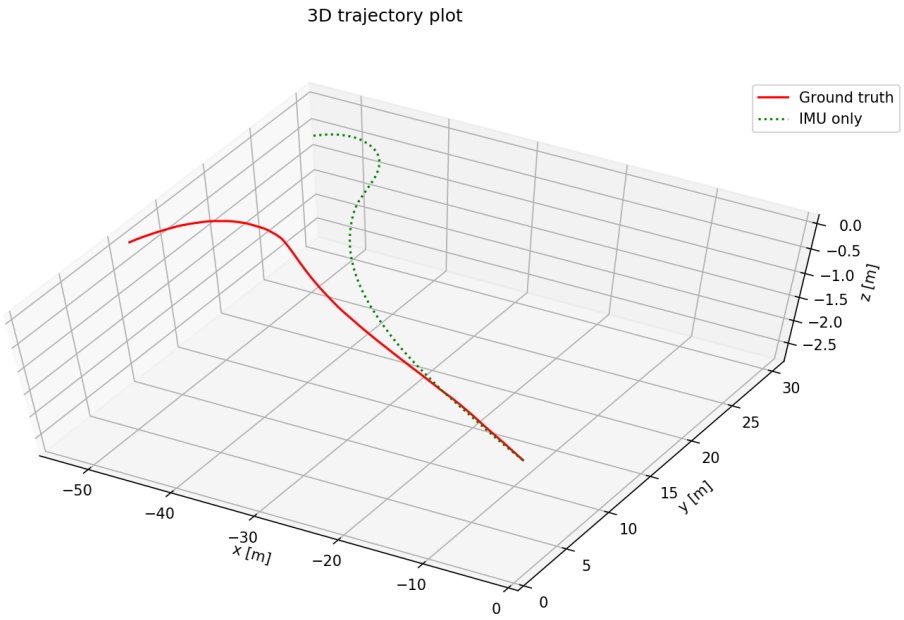

Figure 44: *KITTI* dynamic sequence: *2011_09_26_0035*. 3D trajectory described by ground truth data (solid red) and the output of the filter (dashed green) using inertial data only.

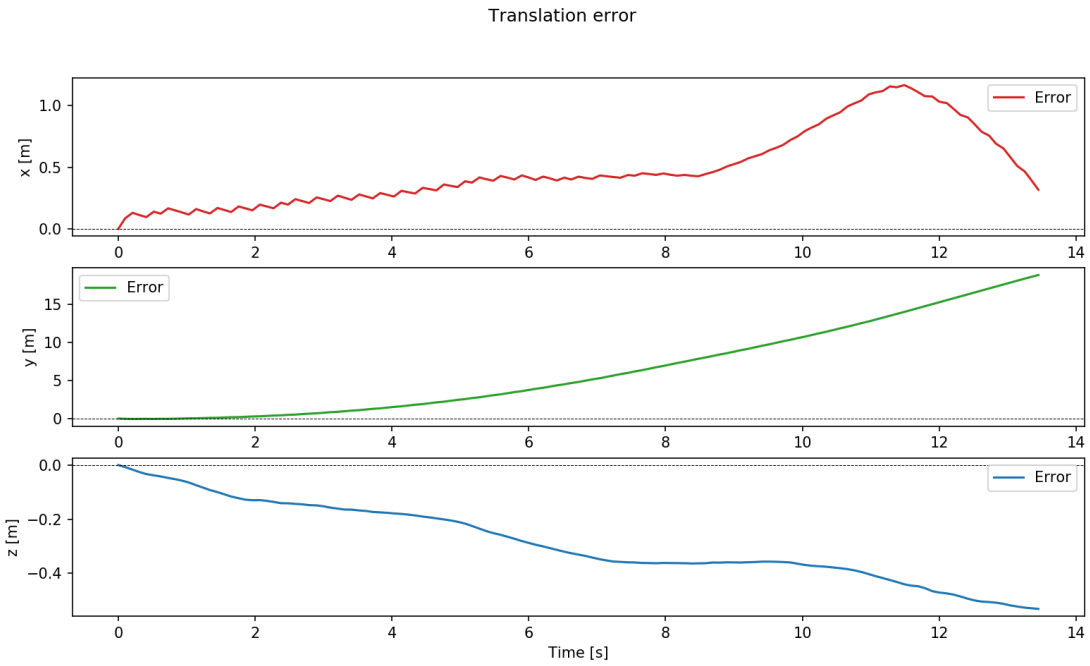

Figure 45: *KITTI* sequence: *2011_09_26_0035*. Errors in X, Y and Z translation estimates $[\mathrm{m}]$ against time $[\mathrm{s}]$ relative to the initial instant, using inertial data only.

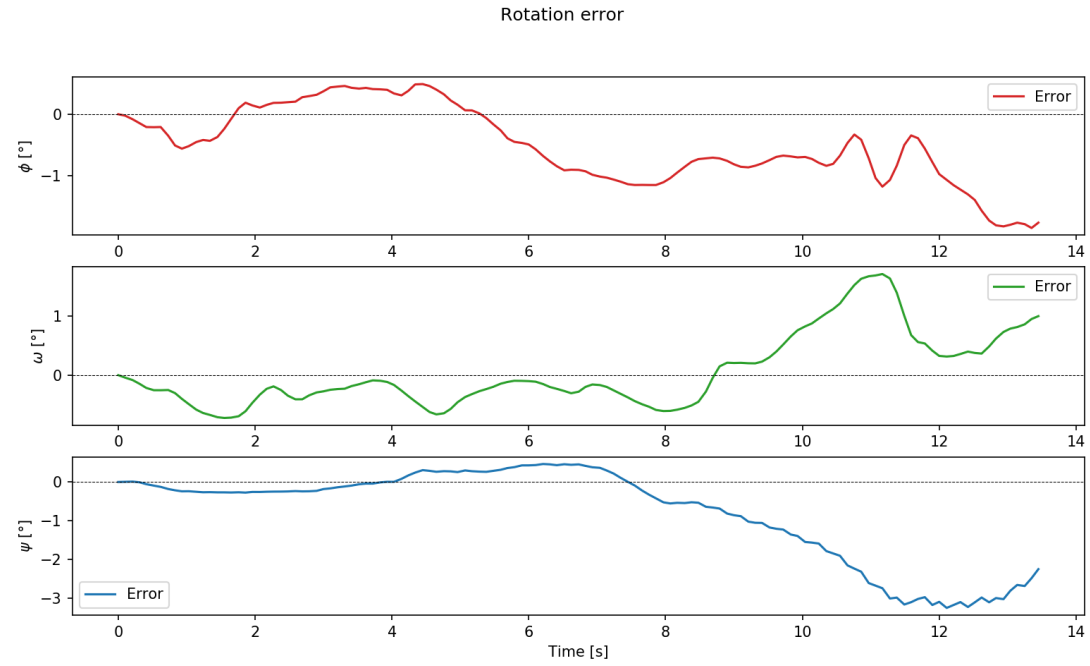

Figure 46: *KITTI* sequence: *2011_09_26_0035*. Error in roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates $[^\circ]$ against time $[\mathrm{s}]$ relative to the initial instant, using inertial data only.

5.3.4 LiDAR-inertial state estimation

The motivation behind the idea of sensor fusion lies in the fact that the mixed nature of these devices is complementary. Focusing on the pair considered in this project, the LiDAR and the IMU, scan matching-based odometry can compensate for the IMU drift, while inertial propagation provides the temporal consistency that instances of cloud registration lack.

Figures 47, 48, 49, 50 and 51 show the results of the sensor fusion in one of the *KITTI* sequences. Two variants have been tested, according to the representation of the measurement covariance: one using the scan registration score, and the other using the inverse of the Hessian matrix computed during the optimization 38.

The NDT hyperparameters set in this experiment coincide with the previous tests in section 5.3.1: voxel size of $\mathbf{1}[\mathbf{m}]$ and an expected outlier ratio of $\mathbf{25}\%$. The convergence criteria was set to a step size $\Delta \boldsymbol{p} = \mathbf{10^{-6}}$ and a maximum of $\mathbf{60}$ iterations.

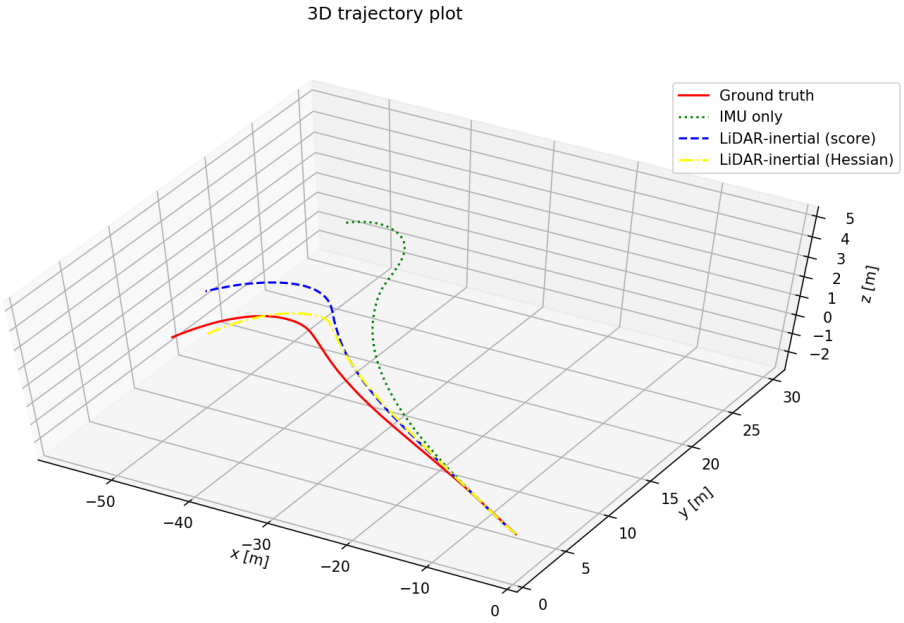

Figure 47: *KITTI* sequence: *2011_09_26_0035*. 3D trajectory described by ground truth data (solid red), inertial-only filter (dashed green) and LiDAR-inertial filter using score-based covariance (dashed blue) and Hessian-based covariance (dashed-dot yellow).

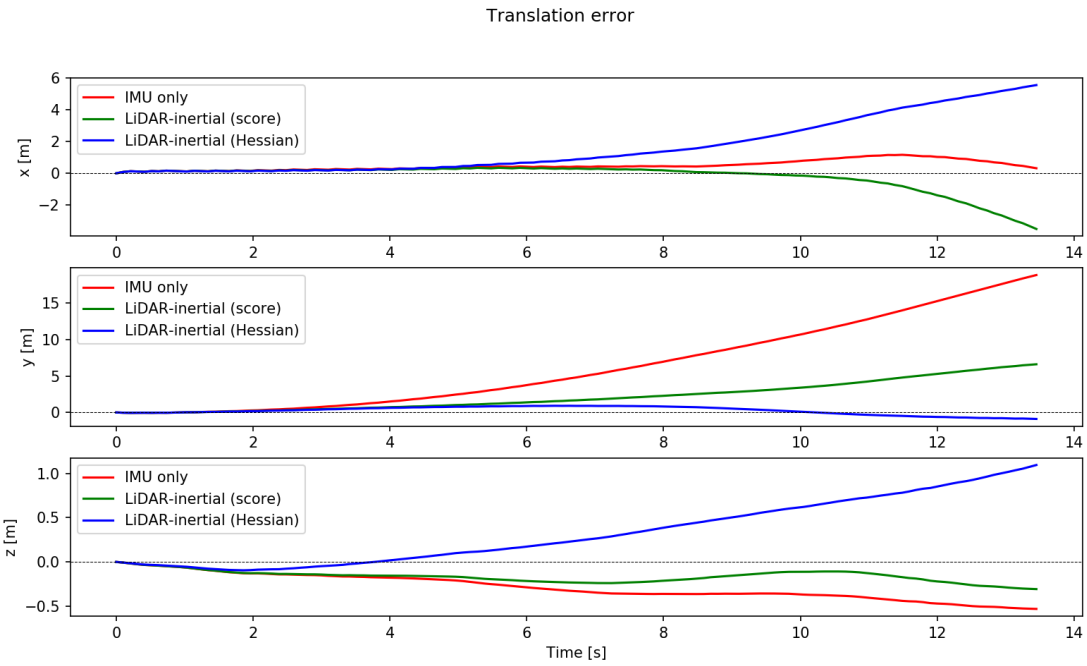

Figure 48: *KITTI* sequence: *2011_09_26_0035*. Errors in X, Y and Z translation estimates $[\mathrm{m}]$ against time $[\mathrm{s}]$ relative to the initial instant, fusing LiDAR and IMU data.

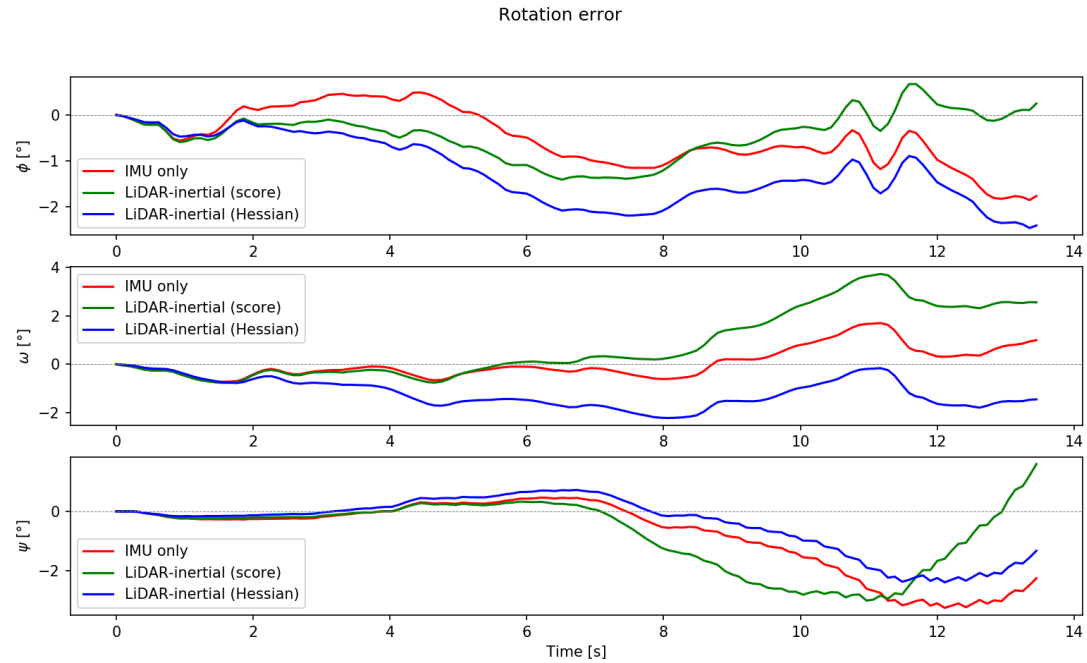

Figure 49: *KITTI* sequence: *2011_09_26_0035*. Error in roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates $[^\circ]$ against time $[\mathrm{s}]$ relative to the initial instant, fusing LiDAR and IMU data.

In terms of the translation components, LiDAR-inertial filters show a mild improvement over the IMU-only version, albeit the similar tendency of the rotation errors for *all* filters (Figure 49) suggests that the NDT doesn't offer much help in correcting the attitude estimates. A remark should be made here: the odometry computed from point cloud registration is relative to the LiDAR local frame, but in order to compute the error states observation, it must be projected to the global reference frame first. This is done through pose composition with the *extrinsics* and the robot's *orientation* as in (74).

Since the latter are part of the state vector, the filter behavior is ill-defined. When the estimated extrinsics and attitude are not accurate, the observations inject error back into the filter. This is hindered by the fact that the motion is constantly *planar*, leading to ambiguous scan alignments (i.e. while X, Y translation and yaw rotation angle are well estimated, Z displacement and roll and pitch angles are noisy).

This means that, even if the rigid motion estimation from scan alignment is exact, the observation in world coordinates comes out erroneously after the change of reference frame (as denoted by Equation 74). The derived errors translate into bias updates that reinforce the filter inertia, inducing in turn a drift-like effect into the LiDAR-based motion estimates.

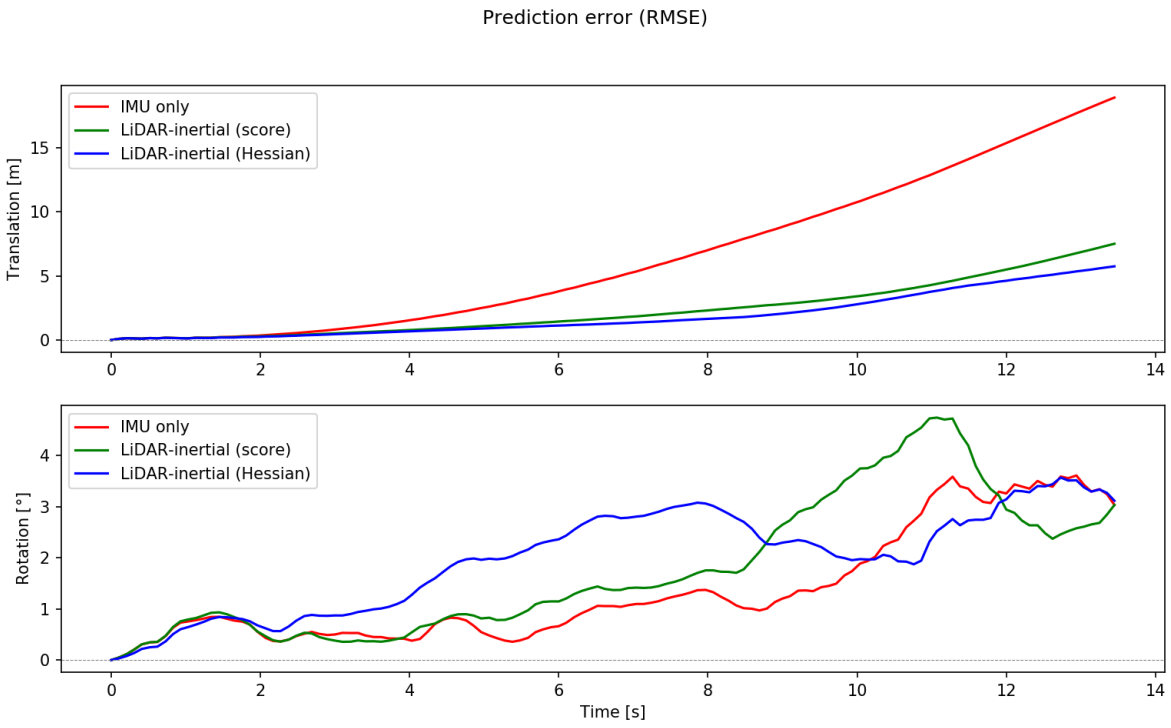

Figure 50: *KITTI* sequence: *2011_09_26_0035*. Root Mean Squared Error (RMSE) of translation [m] and rotation [°] components of rigid motion estimated by IMU-only (red), LiDAR-inertial using score covariance (green) and LiDAR-inertial using inverse Hessian as covariance (blue) filters.

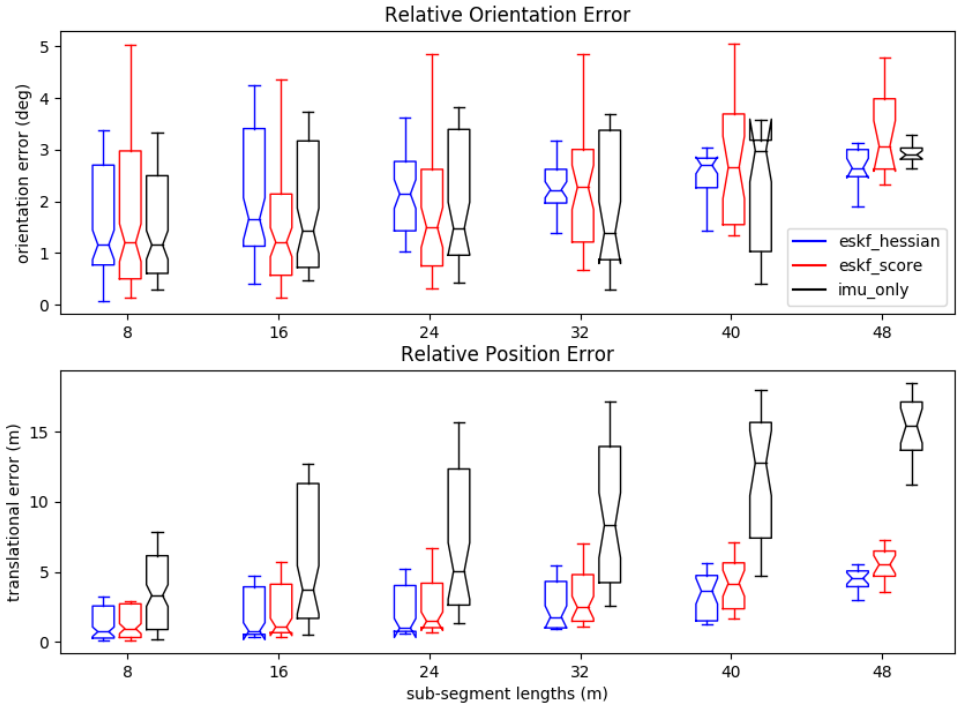

Figure 51: *KITTI* sequence: *2011_09_26_0035*. Relative orientation error [°] and relative translational error [m] with respect to ground truth for different segments of the trajectory, for the IMU-only (black), LiDAR-inertial using score covariance (red) and LiDAR-inertial using inverse Hessian (blue) filters.

Still, Figures 50 and 51 corroborate the improvement of LiDAR-inertial fusion over IMU only filtering. The *RMSE* and *RPE* metrics are lower and more consistent over the duration of the trajectory described by the autonomous platform.

A final note: the implementation of the filtering algorithm does not feature any kind of mapping approach or loop closure technique (hence the designation of odometry or state estimation rather than *SLAM*). Therefore, the drifting can't be systematically suppressed, as there is no global map for the robot to localize itself in.

5.4 State estimation benchmark

Finally, the filtering approach performance will be compared to other state of the art odometry techniques. A brief introduction to their rationale and implementation in publicly available repositories will be given.

5.4.1 LiDAR Odometry And Mapping (LOAM)

Zhang et. al. propose a LiDAR-only technique to solve the simultaneous localization and mapping problem: to subdivide it into two modules running at different frequencies. While a high frequency loop performs coarse odometry using LiDAR features correspondence, a lower frequency one refines the matching through non-linear optimization and registers the point cloud [52].

The features used for motion estimation are edge and planar points, selected in such a way that they are not all clustered in the same region or hardly observable (i.e. in a surface parallel to the laser beam or the boundary of an occluded region). The odometry refinement uses a Levenberg-Marquadt robust fitting method, upon which the point cloud is registered.

The implementation of this SLAM technique is publicly available as a *ROS* package, *loam_velodyne* [115]. The axes convention assumed (X - west, Y - up, Z - north) differs from the recorded data settings (X - north, Y - west, Z - up), so a manual transformation will be applied. Also, the processing node expects the incoming cloud data through a specific *ROS* topic, $/velodyne\_points$, so the name of the recorded topic in the *Rosbags* will be remapped.

5.4.2 LiDAR Inertial Odometry and mapping (LIO-mapping)

Ye et. al. propose a tightly coupled LiDAR-inertial by jointly minimizing a cost function derived from LiDAR and IMU measurements [46], similarly to the factor graph structure proposed in this dissertation for solving the calibration problem (section 3). In fact, IMU pre-integration and LiDAR point-to-plane distances are used as residual terms.

On top of the LiDAR-inertial odometry a refinement method is built to align LiDAR relative features to the global map (as in LOAM). However, they apply a constrained mapping strategy to force the map to always align with gravity, or more precisely, to the Z axis direction (with the implicit assumption that the motion is perpendicular to the Z axis).

This method is implemented in a publicly available *ROS* package, *lio* [116]. As in LOAM, the expected *ROS* topic for cloud data is $/velodyne\_points$, and the IMU processing pipeline is subscribed to the $/imu/data$ topic; the data used for the experiments, recorded in *Rosbag* topics, will then be remapped accordingly.

Since the non-linear optimization proposed uses a fixed-lag smoother and marginalization, some time will pass before the system has enough observations to compute an odometry estimate. Furthermore, IMU measurements in which the excitation is not large enough are automatically discarded, which might result overall in a lower frequency of the odometry updates.

5.4.3 *KITTI* sequences

The performance of the LiDAR-inertial fusion developed in this dissertation will be benchmarked against the state of the art methods discussed previously. Again, point cloud data and inertial measurements from *KITTI* recordings will be used, as well as the ground truth pose of the vehicle to compute error metrics.

The LiDAR-inertial ESKF will consider the inverse Hessian matrix as an the estimate of the rigid motion covariance. The NDT hyperparameters set for this benchmark are a voxel size of $\mathbf{1}[\mathbf{m}]$ and an expected outlier ratio of $\mathbf{5}\%$; the convergence criteria, a step size $\Delta \boldsymbol{p} = \mathbf{10^{-6}}$ and a maximum of $\mathbf{100}$ iterations.

Sequence *2011_09_26_0035* This trajectory has been extensively used during the course of the experiments proposed in this work to showcase the behavior of the ESKF (Figures 39, 44, 47 and more). It features a straightforward motion with a slight curvature to the right, and finishes after a hard turn to the left. Figure 52 collects all estimated trajectories.

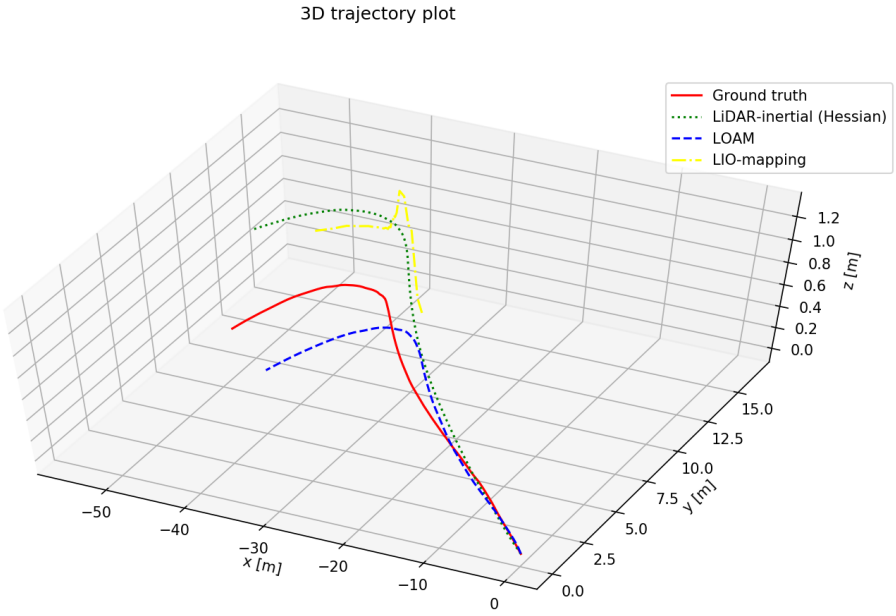

Figure 52: *KITTI* sequence: *2011_09_26_0035*. 3D trajectory described by ground truth data (solid red), ESKF with Hessian-based covariance (dashed green), LOAM (dashed blue) and LIO-mapping (dashed-dot yellow).

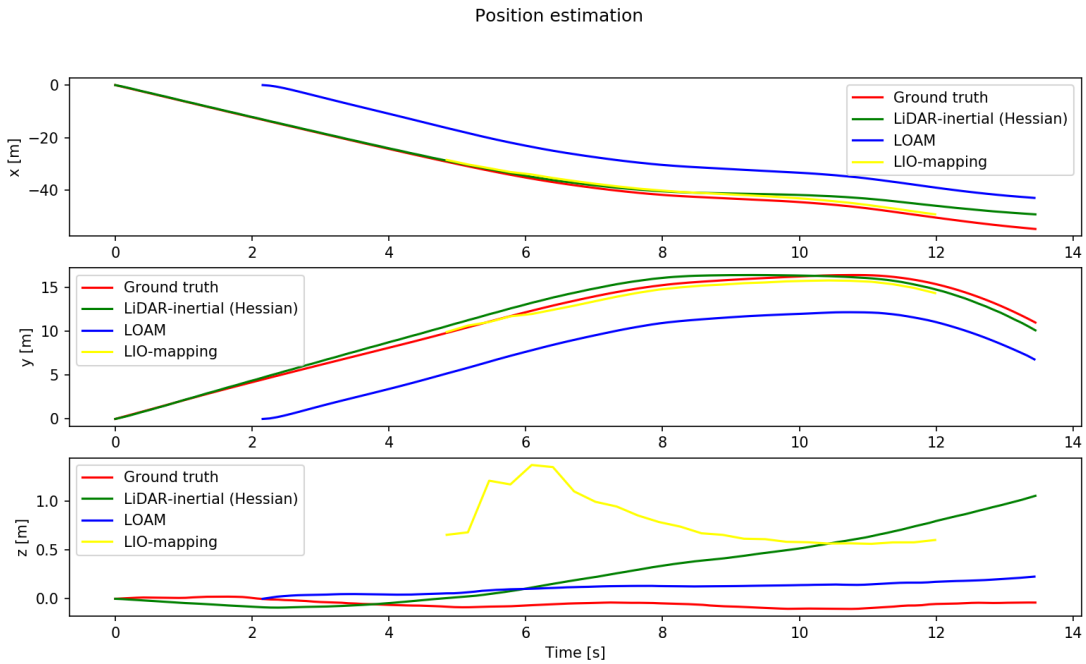

Figure 53: *KITTI* sequence: *2011_09_26_0035*. X, Y and Z translation estimates [m] against time [s] relative to the initial instant.

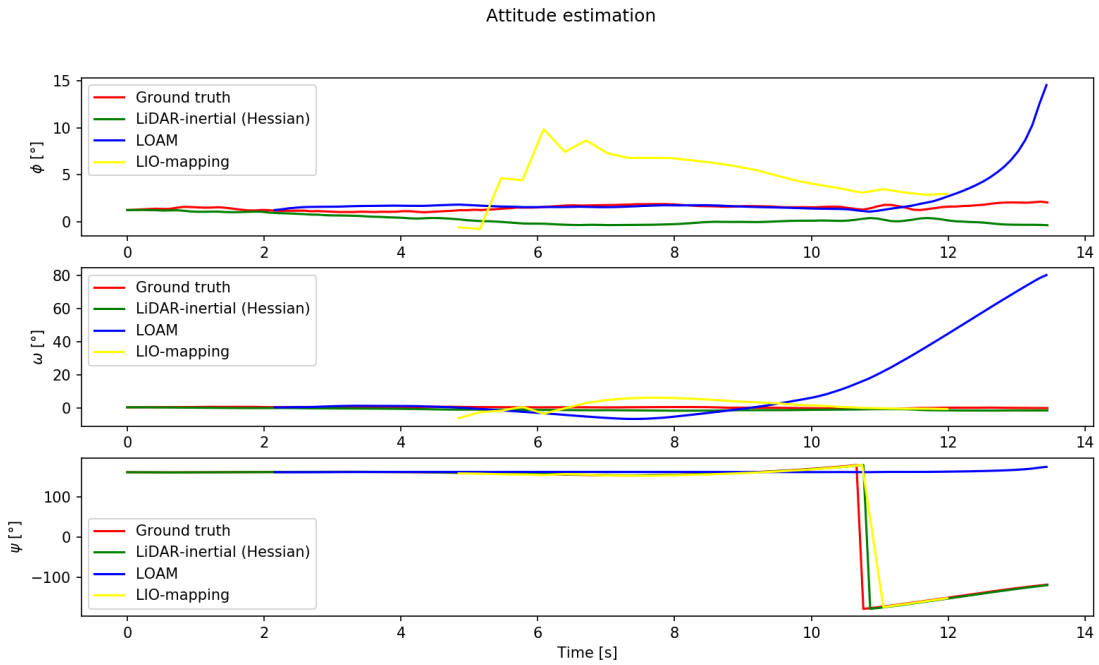

Figure 54: *KITTI* sequence: *2011_09_26_0035*. Roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates $[°]$ against time $[s]$ relative to the initial instant.

In Figures 53 and 54 it is possible to see how LOAM and LIO-mapping need to initially accumulate cloud and IMU data in order to refine their odometry estimates, which explains why there are no predictions at the start of the trajectory. Regardless, both are able to estimate the planar motion accurately $(X, Y$ translations and yaw (ψ) rotations$)$. Note that, even if the plot seems to portray a drastic change in the Z axis orientation, this is because rotation angles are normalized in the $[-180°, +180°]$ interval.

LIO-mapping struggles with the Z alignment, as does the ESKF, and LOAM shows some temporal offset in the estimates. The latter also fails at roll (ϕ) and pitch (ω) rotational alignment (Figure 54), incurring in remarkable drift upon reaching the left turn in the recording (around second 10). This is also captured by the *ROE* metric (Figure 55), which is the largest out of all the methods.

The lower *ROE* and *RPE* metrics associated to the LiDAR-inertal ESKF indicate that it outperforms the other localization techniques. Still, the wide *RPE* box-plots and their tendency to increase as the trajectory progresses suggest that the motion estimates are drifting; while the accuracy that LOAM and LIO-mapping achieve is lower, they are more consistent over time thanks to the odometry refinement step computed with respect to the global map.

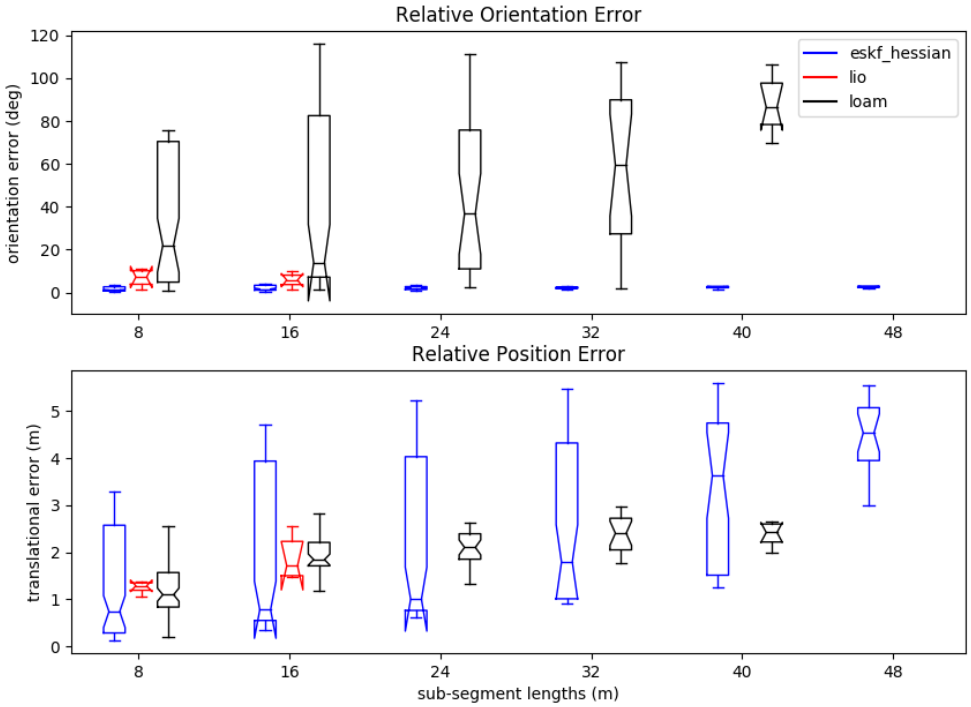

Figure 55: *KITTI* sequence: *2011_09_26_0035*. Relative orientation error (ROE) and relative pose error (RPE) of rigid motion estimated by ESKF with Hessian-based covariance (red), LOAM (green) and LIO-mapping (blue).

Sequence *2011_09_26_0005* This trajectory has been used for previous experiments as well, to showcase the challenges of scan matching in environments with dynamic elements (Figure 36). It features the autonomous vehicle entering and existing a roundabout in an urban environment.

Analyzing Figures 57 and 58 the conclusions are similar to the previous *KITTI* sequence: both LOAM and LIO-mapping estimate X and Y displacements well and struggle with the roll (ϕ) and pitch (ω) rotation prediction. LIO-mapping yaw (ψ) estimate is decent, but the Z alignment diverges from the ground truth; in contrast, LOAM yaw estimation is poor, but nails the Z alignment.

Thus, the ESKF clearly outperforms both LOAM and LIO-mapping, as proved by Figure 59 as well. Its orientation error, quantified by the *ROE* metric, and translational error, or the *RPE* metric, are not only lower than the competition, but also drift less over the course of the trajectory.

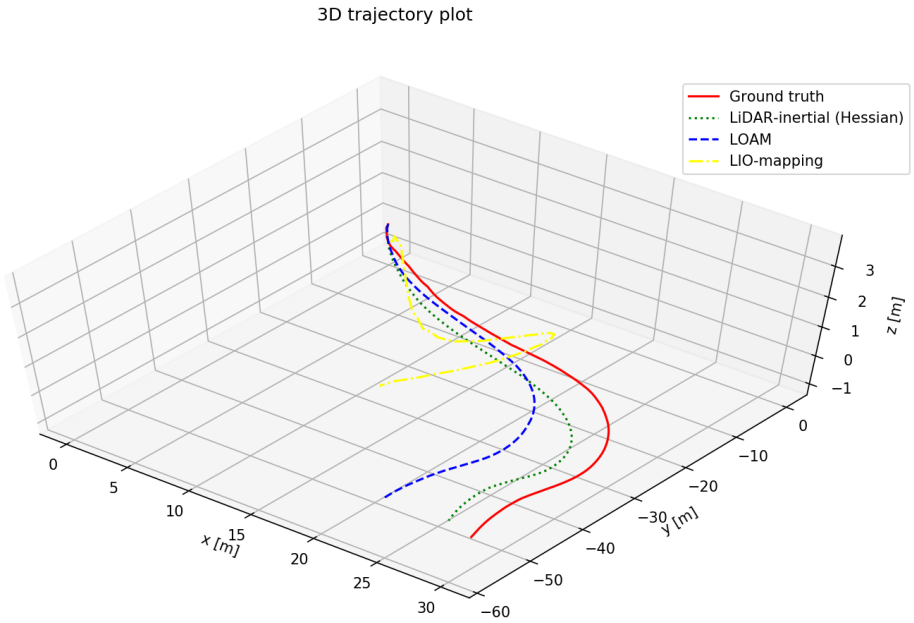

Figure 56: *KITTI* sequence: *2011_09_26_0005*. 3D trajectory described by ground truth data (solid red), ESKF with Hessian-based covariance (dashed green), LOAM (dashed blue) and LIO-mapping (dashed-dot yellow).

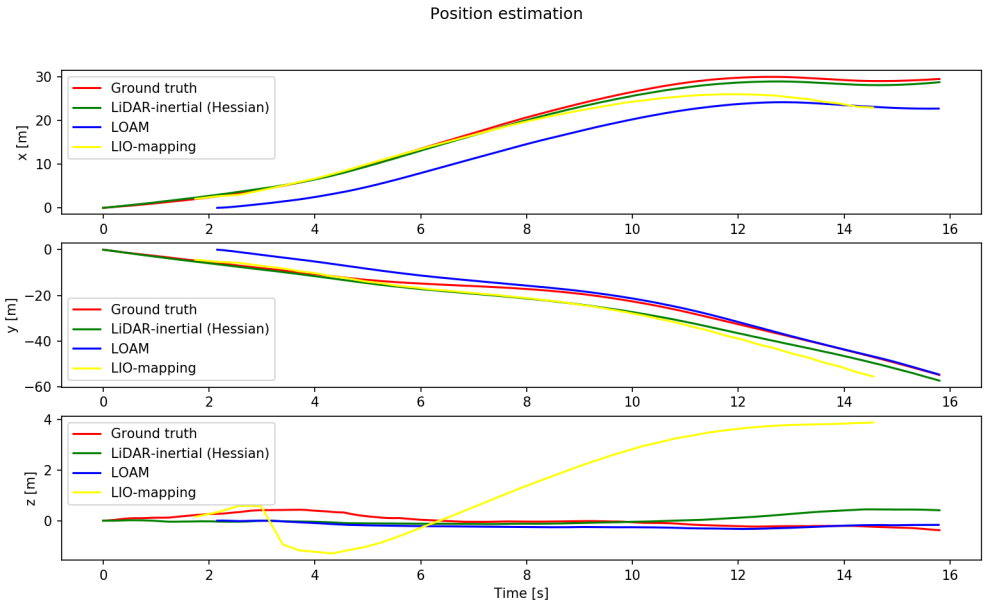

Figure 57: *KITTI* sequence: *2011_09_26_0005*. X, Y and Z translation estimates [m] against time [s] relative to the initial instant.

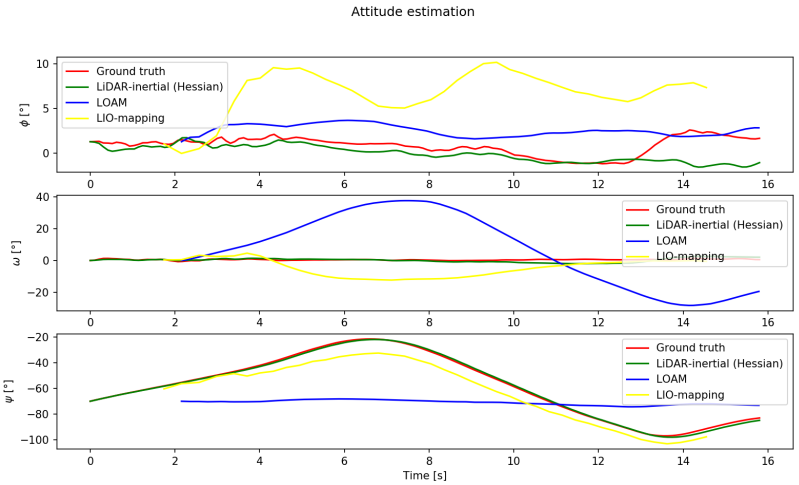

Figure 58: *KITTI* sequence: *2011_09_26_0005*. Roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates [°] against time [s] relative to the initial instant.

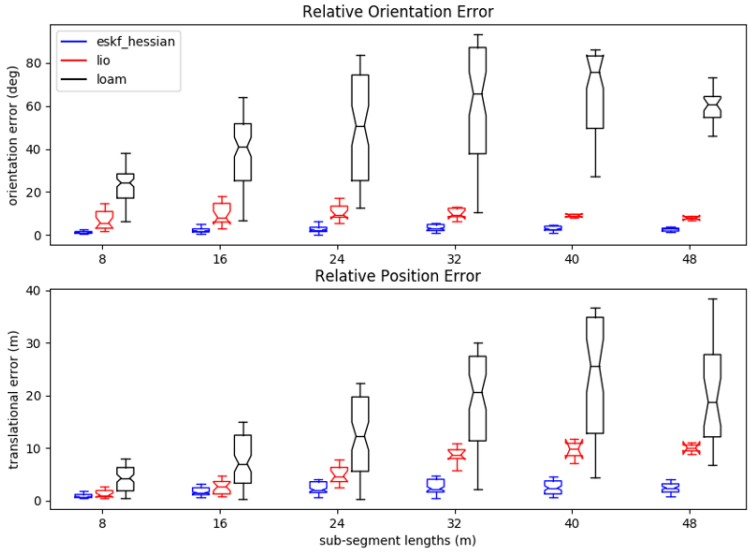

Figure 59: *KITTI* sequence: *2011_09_26_0005*. Relative orientation error (ROE) and relative pose error (RPE) of rigid motion estimated by ESKF with Hessian-based covariance (red), LOAM (green) and LIO-mapping (blue).

Sequence *2011_09_26_0039* The long term consistency of a localization algorithm is essential for the deployment of autonomous mobile platforms. To that end, a longer recording from *KITTI* featuring sustained motion in a straight line is selected, to assess the effects of drift. A tight turn at the beginning of the trajectory will test the rotation estimation capabilities of the algorithms.

As is appreciated in Figure 62 during the first seconds of the trajectory (mainly in the yaw (ψ) angle subplot), the turn causes palpable impact to all localization approaches.

For LOAM, the underestimation of this rotation results in poor alignment with the ground truth, and incurs in heavy positional errors because of this. LIO-mapping estimates an erroneous pitch (ω) angle, resulting in a Z alignment that deviates from the ground truth over time. Finally, while the ESKF is somewhat capable to cope with the yaw angle evolution, roll (ϕ) and pitch estimates show unstable behavior (which translates coherently into error in positional alignment in Figure 61).

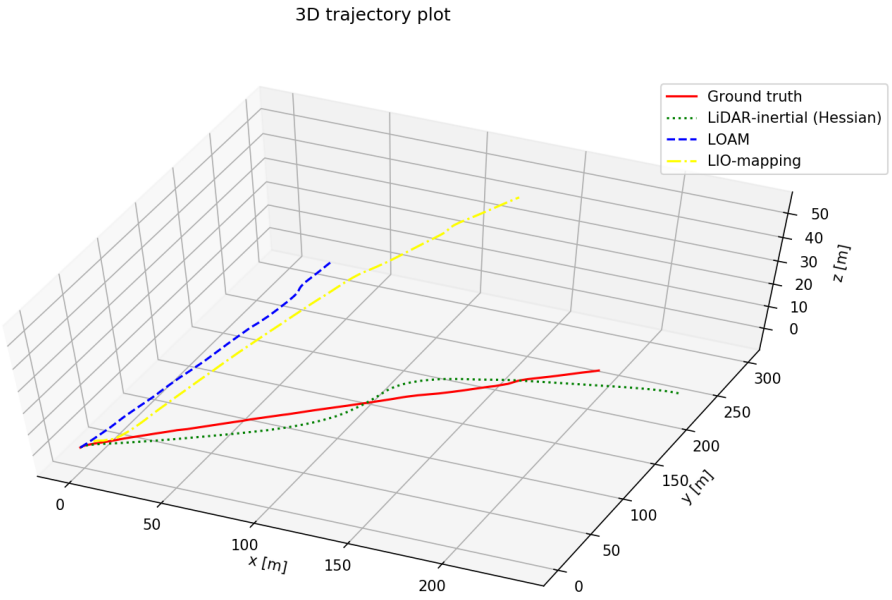

Figure 60: *KITTI* sequence: *2011_09_26_0039*. 3D trajectory described by ground truth data (solid red), ESKF with Hessian-based covariance (dashed green), LOAM (dashed blue) and LIO-mapping (dashed-dot yellow).

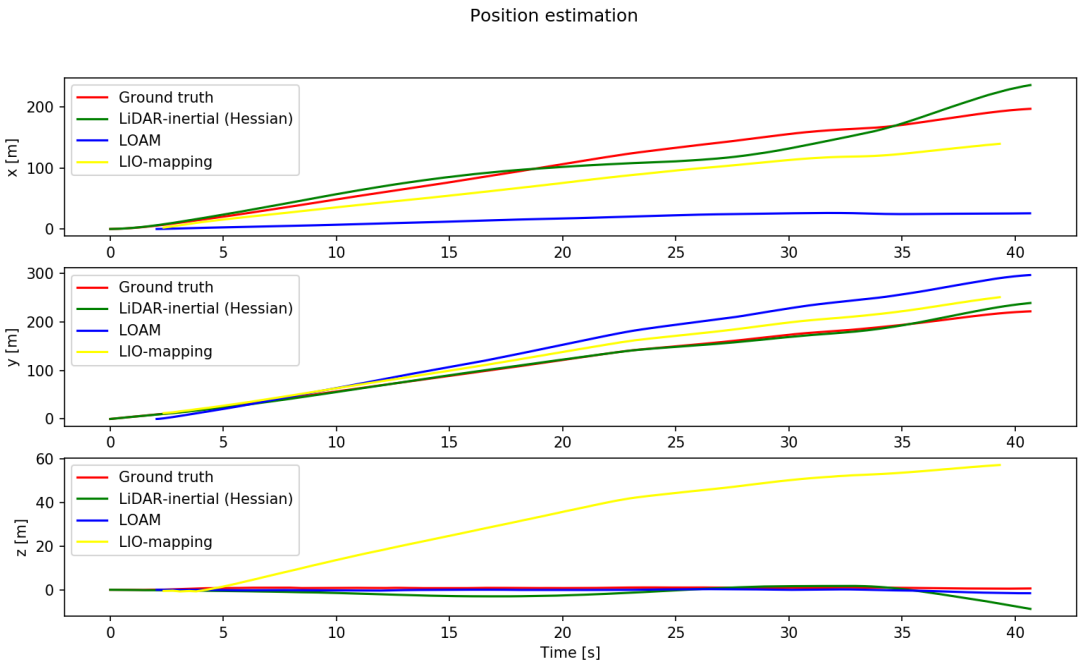

Figure 61: *KITTI* sequence: *2011_09_26_0039*. X, Y and Z translation estimates [m] against time [s] relative to the initial instant.

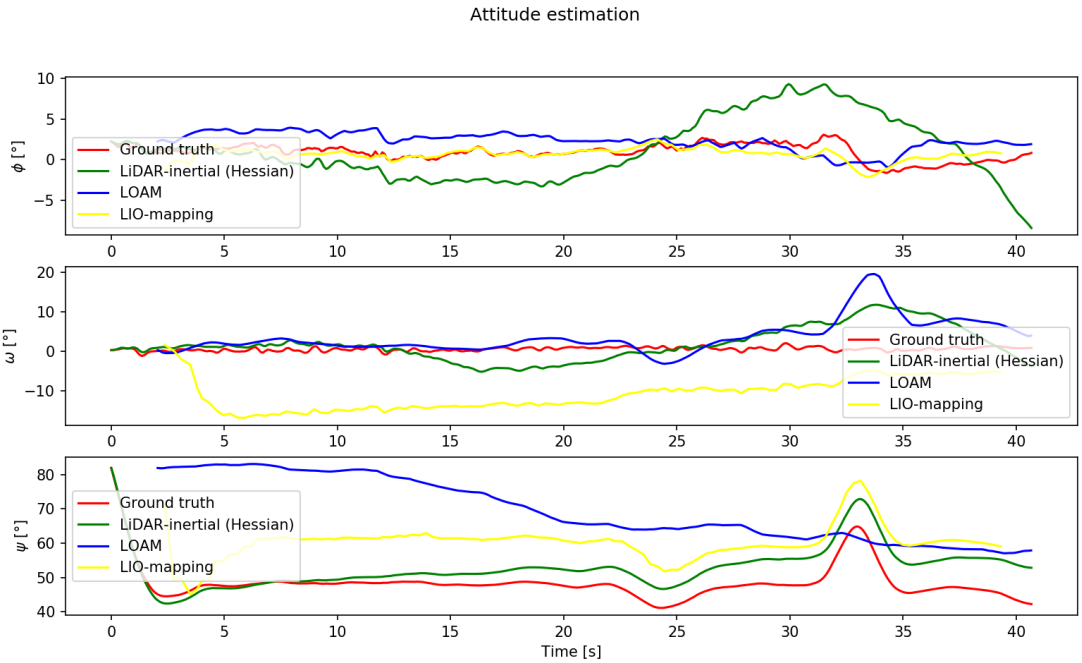

Figure 62: *KITTI* sequence: *2011_09_26_0039*. Roll (ϕ), pitch (ω) and yaw (ψ) rotation estimates [°] against time [s] relative to the initial instant.

However, there is a key difference between the filter and the other approaches. From visual inspection of Figure 60 it is possible to discern how, even if the rotational alignment of LOAM and LIO-mapping fails at the beginning, their motion estimates are consistent over time (as denoted by the 'S' turn correctly captured in the $[30, 35][s]$ interval). In contrast, while the ESKF is able to track this maneuver in terms of yaw rotation, the roll and pitch estimates start accumulating error, thus causing the position estimation to drift away from the ground truth.

Figure 63 further proves this point: the *ROE* metric identifies how the ESKF orientation error grows over the course of the trajectory, while LOAM and more specifically LIO-mapping are able to keep that drift somewhat over control (even if in average terms the magnitudes of the errors are a comparable mean value). The *RPE* metric is even clearer at portraying the ESKF drifting by its increasing average and wider box-plot; LIO-mapping showcases the best performance in this area, in terms of accuracy and temporal consistency.

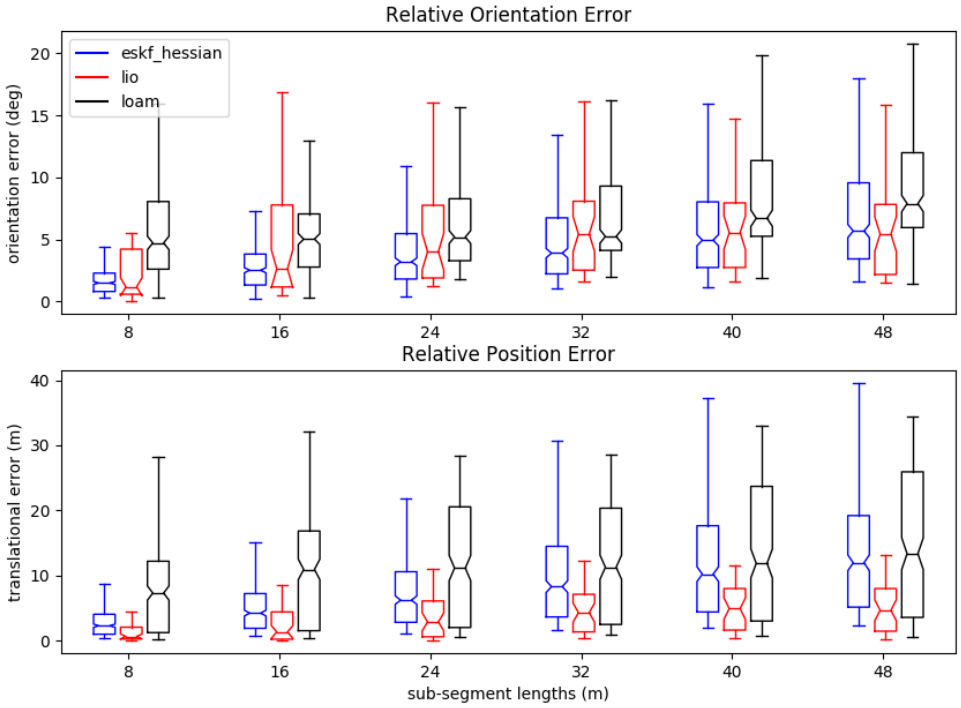

Figure 63: *KITTI* sequence: *2011_09_26_0039*. Relative orientation error (ROE) and relative pose error (RPE) of rigid motion estimated by ESKF with Hessian-based covariance (red), LOAM (green) and LIO-mapping (blue).

5.5 Benchmark summary

The relative position and orientation error of the three localization algorithms contemplated in the benchmarking are summarized in this section, in a table per experiment.

	$\overline{\mathbf{RPE}}_{\mathbf{8m}}$	$\overline{\mathbf{RPE}}_{\mathbf{16m}}$	$\overline{\mathbf{RPE}}_{\mathbf{24m}}$	$\overline{\mathbf{RPE}}_{\mathbf{32m}}$	$\overline{\mathbf{RPE}}_{\mathbf{40m}}$	$\overline{\mathbf{RPE}}_{\mathbf{48m}}$
ESKF	**0.7376**	**0.7767**	**1.0018**	**1.7926**	3.6335	**4.5382**
LOAM	1.0972	1.8435	2.1028	2.4029	**2.4205**	—
LIO-mapping	1.2858	1.7117	—	—	—	—

Table 1: Average relative position error $[m]$ in *KITTI* sequence *2011_09_26_0035* for ESKF, LOAM and LIO-mapping.

	$\overline{\mathbf{ROE}}_{\mathbf{8m}}$	$\overline{\mathbf{ROE}}_{\mathbf{16m}}$	$\overline{\mathbf{ROE}}_{\mathbf{24m}}$	$\overline{\mathbf{ROE}}_{\mathbf{32m}}$	$\overline{\mathbf{ROE}}_{\mathbf{40m}}$	$\overline{\mathbf{ROE}}_{\mathbf{48m}}$
ESKF	**1.1598**	**1.6620**	**2.1430**	**2.2248**	**2.6968**	**2.6431**
LOAM	22.0465	13.9340	36.6980	59.4695	86.4060	—
LIO-mapping	7.2559	5.7795	—	—	—	—

Table 2: Average relative orientation error $[°]$ in *KITTI* sequence *2011_09_26_0035* for ESKF, LOAM and LIO-mapping.

The ESKF incurring in the lowest relative position and orientation errors out of all the benchmarked localization techniques (Tables 1 and 2) shows a level of performance comparable to state of the art solutions. The increasing error as the trajectory progresses denotes a certain drift, which is coherent with the lack of a loop closure methodology.

	$\overline{\mathbf{RPE}}_{\mathbf{8m}}$	$\overline{\mathbf{RPE}}_{\mathbf{16m}}$	$\overline{\mathbf{RPE}}_{\mathbf{24m}}$	$\overline{\mathbf{RPE}}_{\mathbf{32m}}$	$\overline{\mathbf{RPE}}_{\mathbf{40m}}$	$\overline{\mathbf{RPE}}_{\mathbf{48m}}$
ESKF	**0.7484**	**1.5327**	**2.0548**	**2.2681**	**2.4023**	**2.4484**
LOAM	4.2703	7.0509	12.2787	20.6511	25.6612	18.7603
LIO-mapping	1.0597	2.7120	4.6690	8.6925	9.8252	10.0267

Table 3: Average relative position error $[m]$ in *KITTI* sequence *2011_09_26_0005* for ESKF, LOAM and LIO-mapping.

	$\overline{\mathbf{ROE}}_{\mathbf{8m}}$	$\overline{\mathbf{ROE}}_{\mathbf{16m}}$	$\overline{\mathbf{ROE}}_{\mathbf{24m}}$	$\overline{\mathbf{ROE}}_{\mathbf{32m}}$	$\overline{\mathbf{ROE}}_{\mathbf{40m}}$	$\overline{\mathbf{ROE}}_{\mathbf{48m}}$
ESKF	**1.1753**	**1.9083**	**2.3891**	**3.0043**	**2.7808**	**2.8827**
LOAM	24.2714	40.9271	50.8425	65.6741	75.8382	60.6648
LIO-mapping	5.6910	8.1120	9.4885	9.4832	9.0958	7.9829

Table 4: Average relative orientation error [°] in *KITTI* sequence *2011_09_26_0005* for ESKF, LOAM and LIO-mapping.

In the second experiment, the Normal Distributions Transform shows superior robustness in dealing with dynamic scenes, as denoted by the significantly lower *RPE* and *ROE* metrics in Tables 3 and 4. This is explained by the fact that the scan matching odometry does not require explicit feature correspondences.

	$\overline{\mathbf{RPE}}_{\mathbf{8m}}$	$\overline{\mathbf{RPE}}_{\mathbf{16m}}$	$\overline{\mathbf{RPE}}_{\mathbf{24m}}$	$\overline{\mathbf{RPE}}_{\mathbf{32m}}$	$\overline{\mathbf{RPE}}_{\mathbf{40m}}$	$\overline{\mathbf{RPE}}_{\mathbf{48m}}$
ESKF	2.2658	4.2073	6.2033	8.2762	10.1894	11.9348
LOAM	7.2859	10.9135	11.1267	11.2561	11.8490	13.3355
LIO-mapping	**0.6245**	**1.3100**	**2.8572**	**4.2786**	**4.8980**	**4.6417**

Table 5: Average relative position error [m] in *KITTI* sequence *2011_09_26_0039* for ESKF, LOAM and LIO-mapping.

	$\overline{\mathbf{ROE}}_{\mathbf{8m}}$	$\overline{\mathbf{ROE}}_{\mathbf{16m}}$	$\overline{\mathbf{ROE}}_{\mathbf{24m}}$	$\overline{\mathbf{ROE}}_{\mathbf{32m}}$	$\overline{\mathbf{ROE}}_{\mathbf{40m}}$	$\overline{\mathbf{ROE}}_{\mathbf{48m}}$
ESKF	1.4951	**2.5212**	**3.1991**	**3.8973**	**4.9421**	5.6807
LOAM	4.7175	5.0312	5.1905	5.2405	6.7056	7.8867
LIO-mapping	**1.1830**	2.6416	4.0177	5.4336	5.5506	**5.4307**

Table 6: Average relative orientation error [°] in *KITTI* sequence *2011_09_26_0039* for ESKF, LOAM and LIO-mapping.

Finally, the third experiment metrics denote a state-of-the-art-level performance from the ESKF, but most importantly, highlight the main drawback of the filtering algorithm: its lack of long term consistency. This is clearly appreaciated in the relative orientation error (Table 6), where the ESKF initially commits a smaller error in comparison to LIO-mapping, but the latter is able to keep it from drifting, showing a lower *ROE* by the end of the trajectory.

CHAPTER 6

Socioeconomic analysis

The rise of innovation-led start-ups and the global market needs for process automation in fields like construction site monitoring [117], warehouse logistics [118; 119] or agriculture [120] denote an increasingly closer relationship between research and business in the field of autonomous robotics. The matureness level of the technology has reached a point in which it is viable to deploy robotic solutions in particular applications.

A key aspect of sensors in the market of mobile robotics is that the wide assortment available of exteroceptive and proprioceptive sensors are not mutually exclusive, but rather *complementary*. Cameras, inertial measurement units, LiDARs, radars, ... all have their pros and cons regarding range, size, robustness to illumination conditions, availability of color and depth information, etc. but their fusion in multi-sensor configurations manages to exploit their strengths [121].

The international LiDAR market is an example of this. It was estimated at around 1 billion American dollars (1 bn USD) in 2018 with a significant expected annual growth rate of 35% up to 2025 [122]. Regardless of how optimistic the forecast may be, it is apparent in qualitative terms that the LiDAR is here to stay. The numerous references contained in this dissertation back up this claim providing evidence of the remarkable attention that this sensor has drawn in multiple lines of investigation.

The amount of stakeholders in the LiDAR market is also growing, broadening the variety of options for customers and steadily dropping the final product cost. The diversity of applications in which the LiDAR is involved is a testament to its flexibility and robustness as a perception device [123], and technologies like autonomous driving could be powered by their capabilities.

If the automation trends were to evolve as they are currently, the potential social impact would be remarkable [121]. There could be a paradigm shift in the job market, where humans performing repetitive and tedious tasks could be replaced by a robot, or collaborative robots could be employed in their assistance. In any case, the introduction of robots in the workplace could result in a more specialized workforce and new human-robot interactions to be contemplated (and regulated if necessary).

Furthermore, a lot of topics regarding ethics in robotics, biasing in artificial intelligence systems or sensitive issues in social robotics [124] remain to be discussed. The outcome of this debate may backlash, causing a negative social response to autonomous systems. However, there is not enough data to safely predict the impact and presence of autonomous mobile robotics in the long run, so tracking the evolution of this debate will be a key factor to the rise (or fall) of these technologies.

6.1 Cost analysis

In order to sketch a budget for this project, the solution that will be taken into account for the cost analysis is the one presented in section 5.1.1, since it is the hardware platform that has been used to collect real data (disregarding the data retrieved from publicly available datasets).

The robotic platform itself, named *Jackal*, is manufactured by *Clearpath Robotics*. Its estimated price is **15000 €**, including tires, chassis, drive train, an internal GNSS/IMU and built-in *ROS* functionality. The inertial measurement unit, the industrial-grade model *Microstrain 3DM-GX5-45* by *LORD Sensing Systems*, is valued at around 3000$ ($\approx$ **2700 €**). The LiDAR sensor is manufactured by *Velodyne*, model *VLP-16*, and its retail price sits at 4000$ ($\approx$ **3500 €**)

Figure 64: Jackal by Clearpath Robotics (left) [105] and VLP-16 LiDAR by Velodyne (right) [107].

In order to have an estimate of the time dedicated to the engineering project, the *30 ECTS* credits recognized by this dissertation have been translated into working hours using a factor of *25h/ECTS* (applicable in Spain), yielding a total of ***750 hours***. Under the hypothesis of a *10€/h* salary of a junior robotics engineer, an estimated budget for the project is computed as follows:

$$\begin{aligned} \mathbf{B}(\mathbf{C}, \mathbf{t}) = C_{ROB} + C_{IMU} + C_{LID} + t\, C_{ENG} = \\ 15000 + 2700 + 3500 + 750 * 10 = \mathbf{28700\,€} \end{aligned} \tag{75}$$

Figure 65: Microstrain *3DM-GX5-45* IMU by LORD Sensing Systems [108].

CHAPTER 7

Environmental analysis

Although the software developed in this dissertation for calibration and state estimation does not intrinsically have any direct implications on the environment, the adoption of autonomous robots does. For one, it encourages the use of electric motors in detriment of gasoline engines, which have been shown to be less polluting during operation and modestly compatible with renewable energy sources for battery charging [125]. It should be noted that the overall contribution to the carbon footprint in terms of CO_2 emissions depends on their lifetime, since the production of batteries is highly contaminant.

In the end, the implications of autonomous solutions are application-specific. An example that has generated a lot of debate is the environmental friendliness of self-driving cars; while they could be programmed to optimize route efficiency, reduce congestion in densely populated areas and promote car-sharing, they could also indulge society in using private transportation more frequently and for longer distances, resulting in an increased energy consumption [126; 127]. On another note, automated monitoring systems could help reduce the environmental impact caused by preventable accidents (e.g. defective assemblies, leaks of contaminating substances, degraded materials...) thanks to early detection [128].

A final remark is made about health safety: the laser-based operation principle of LiDAR technology has raised some concerns regarding the aggressiveness of certain wavelengths on the human retina [129]. LiDAR manufacturers are looking to meet safety standards in response and are considering this fact in their designs, so that the technology is as non-invasive as possible. In what regards to IMU's, their proprioceptive nature implicitly ensures their non-invasiveness, which guarantees safe operation around humans and wildlife.

CHAPTER 8

Conclusions

Multi-sensor calibration is a challenging endeavor. It requires specific knowledge about the particular devices at hand in order to formulate the problem, and about the software tools necessary to solve it. Furthermore, the balance between accuracy requirements, computational load and ease of calibration procedure has to be taken into account in the first place.

The implementation of the batch optimization algorithm for calibration has proven to be remarkably complex. It integrates various modules that deserve a thorough study by themselves (pre-integration theory, Gaussian Processes, RANSAC, ...) and results in a non-linear optimization problem with dynamically weighted residuals, which is hard to solve without an already somewhat precise initial estimate.

This complexity of the calibration problem, together with the limitations in regards to the acquisition of ground truth data for the parameters that are aimed to be optimized, have difficulted the analysis of the results obtained. However, the objectives set at the beginning of the project, as presented in section 1.1, have been achieved.

On the second stage of the project, the LiDAR-inertial state estimation, the accurate calibration of IMU biases and IMU-LiDAR extrinsic parameters has made its influence apparent. For all the robustness that the Normal Distributions Transform has shown in the proposed experiments, even in mildly dynamic environments, the observations computed within the filtering framework are still prone to errors due to bad calibration and orientation estimates.

Indeed, the benchmarking between the Error State Kalman Filter and other state of the art localization techniques such as LOAM and LIO-mapping has proven the ESKF performance to be on a similar level. However, it has also pointed out one of the main weaknesses of the filtering algorithm: its lack of long term consistency. The author believes that the implementation of a loop closure layer is a high priority task towards the compensation of this drawback, as it is essential to systematically eliminate drift.

On that note, the experiments on localization have shown the importance of accurate attitude estimation. Incorrect gravity alignment and scan matching observations tainted by orientation errors degrade the performance of the filter. The superficial study on Lie theory has also enabled the author to identify the fact that large orientation errors parameterized in the Cartesian space $\Delta\theta \in \mathbb{R}^3$ do not accurately represent members of the rotation group $\mathcal{SO}(3)$. In the literature, this recently has lead to the proposal of *invariant Kalman Filters* [130; 131], whose aim is to represent the state vector as a member of the manifold and perform filtering there, without the need for mapping to other spaces.

On the other hand, the relative easiness of implementation of the ESKF has allowed the evaluation of different solutions (NDT hyperparameter tuning, weighted averaging for on-manifold uncertain rigid motions, multiple characterizations of the NDT measurement covariance). Finally, it is possible to conclude that the objectives set at the beginning of this dissertation were met. However, there is a caveat: it was not possible to test the online performance of the algorithm due to limited access to the robotic platform, which is briefly discussed in the following section.

Limitations

Due to the sanitary alarm caused by a worldwide spread pandemic, the majority of the time period allocated to the development of the Master's Thesis has been relegated to teleworking. This has negatively impacted the capacity of the author to interact with the robotic platform presented in section 5.1.1. As a result, the quantity of data captured for experiments has been limited.

This situation has also made online testing of the state estimation implementation impossible. While offline experiments using *Rosbag* files with recorded data suggest that real-time operation is feasible, no further proof has been gathered. Admittedly, this could involve code refactoring for performance optimization, as running the software on real hardware in real time is a challenge in itself.

Future work

Several routes for future work can be derived from the conclusions drawn on this project. First of all: in order to deal with the lack of long term consistency of the filtering algorithm, the development of a loop closure technique could be of interest. Enabling the autonomous mobile platform to localize itself with respect to a map of the environment, potentially profiting from the Normal Distributions Transform continuous probabilistic treatment, would mitigate the drifting effect caused by erroneous attitude estimates.

Another possible fix for that problem could be to develop the filtering algorithm directly on the manifold. This would grant higher stability to the orientation error predictions and updates, making the error dynamics invariant (hence the name of *invariant* filters) to the trajectory.

In order to further validate the results obtained through the offline calibration procedure, running the software on more instances of data could be of interest. A deeper analysis on the factor graph structure could help in selecting the most adequate solver for this kind of large, sparse problem, thus improving its convergence and reducing its required processing time.

On the long run, the work developed in this dissertation lends itself to the consideration of other multi-sensor configurations. One of special interest could be the incorporation of a camera to the LiDAR-IMU pair, resulting in LiDAR-visual-inertial fusion. There is remarkable potential in this set-up, and much room for investigation on how to formulate a three-way calibration and exploit the information that these sensors are able to gather.

Acknowledgements

I am deeply indebted to *Scaled Robotics* for granting me the opportunity to work on a state-of-the-art topic within the robotics community that can provide value to a real-world application. The support and guidance I have received from its team, and specially from my mentors, Bharath Sankaran and Pol Cirujeda, have resulted in an invaluable source of personal and professional growth, both in the high-level perspective and in the small details.

I would like to express my sincere thanks to my academic advisor, Juan Andrade, for providing me with informed, actionable feedback during the development of the project, as well as with logistic support. Likewise, I would like to thank Alberto Sanfeliu for accepting to be the secretary of this Master's Thesis. I would also like to extend the appreciation to Joan Solà for sharing his deep understanding of geometry and optimization.

Last but not least, I would like to stress the endless support I have received from my family. Thanks to my parents Mariano and Lola and my brother Enrique, who have taught me the value of empathy and effort, and have raised me to be the person I am today. And specially to my other half, Ana Belén, who makes me strive to be the best version of myself.

References

[1] H. Ahmed and H. M. La, "Education-robotics symbiosis: An evaluation of challenges and proposed recommendations," in *2019 IEEE Integrated STEM Education Conference (ISEC)*, pp. 222–229, 2019.

[2] R. H. Taylor, "A perspective on medical robotics," *Proceedings of the IEEE*, vol. 94, no. 9, pp. 1652–1664, 2006.

[3] A. Tapus, M. J. Mataric, and B. Scassellati, "Socially assistive robotics [grand challenges of robotics]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 35–42, 2007.

[4] M. Schnaubelt, S. Kohlbrecher, and O. von Stryk, "Autonomous assistance for versatile grasping with rescue robots," in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 210–215, 2019.

[5] D. B. O. Boesl and B. Liepert, "4 robotic revolutions - proposing a holistic phase model describing future disruptions in the evolution of robotics and automation and the rise of a new generation 'r' of robotic natives," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1262–1267, 2016.

[6] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[7] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[8] M. Rickert, A. Sieverling, and O. Brock, "Balancing exploration and exploitation in sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1305–1317, 2014.

[9] T. B. Chistyakova, I. G. Kornienko, and I. V. Novozhilova, "Computer system for nanostructured ceramic materials quality control," in *2017 IEEE II International Conference on Control in Technical Systems (CTS)*, pp. 9–12, 2017.

[10] Y. Pan, H. Yu, and P. Jiang, "Application of intelligent control in raw meal quality control," in *2019 Chinese Control And Decision Conference (CCDC)*, pp. 2514–2519, 2019.

[11] H. M. Taylor, C. Dondrup, and K. S. Lohan, "Introducing a scalable and modular control framework for low-cost monocular robots in hazardous environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6421–6426, 2019.

[12] X. Li, W. Yu, X. Lin, and S. S. Iyengar, "On optimizing autonomous pipeline inspection," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 223–233, 2012.

[13] M. S. Bin Mohamad Soberi, Z. H. Ismail, M. Z. Bin Zakaria, and K. Sammut, "Autonomous ship hull inspection by omnidirectional path and view," in *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pp. 38–43, 2016.

[14] M. Adams, W. S. Wijesoma, and A. Shacklock, "Autonomous navigation: Achievements in complex environments," *IEEE Instrumentation Measurement Magazine*, vol. 10, no. 3, pp. 15–21, 2007.

[15] M. Khodabandeh and A. Mohammad-Shahri, "Experimental study of uncertainty measures with sensor fusion techniques," in *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*, vol. 2, pp. 151–154, 2010.

[16] M. K. Nutalapati, L. Arora, A. Bose, K. Rajawat, and R. M. Hegde, "A generalized framework for autonomous calibration of wheeled mobile robots," 2020.

[17] J. Shepherd, D. James, H. Espinosa, D. Thiel, and D. Rowlands, "A literature review informing an operational guideline for inertial sensor propulsion measurement in wheelchair court sports," *Sports*, vol. 6, p. 34, 04 2018.

[18] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation.," in *Robotics: Science and Systems* (L. E. Kavraki, D. Hsu, and J. Buchli, eds.), 2015.

[19] J. C. Chow, "Statistical sensor fusion of a 9-dof mems imu for indoor navigation," *arXiv preprint arXiv:1802.04388*, 2018.

[20] B. L. Jose, A. H. Ballado, C. A. E. Robas, J. T. Orias, R. A. Haga, and S. A. P. Bentir, "Validation of lidar point clouds for classification of high-value crops using geometric-and reflectance-based extraction algorithm," in *2017 IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pp. 1–6, 2017.

[21] C. Barbanson, C. Mallet, A. Gressin, P. Frison, and J. Rudant, "Fusion of lidar and radar data for land-cover mapping in natural environments," in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3715–3718, 2015.

[22] C. Chiu, L. Fei, J. Liu, and M. Wu, "National airborne lidar mapping and examples for applications in deep-seated landslides in taiwan," in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 4688–4691, 2015.

[23] K. Sanjaya, F. Henning, and K. R. Purba, "3d lidar city model application and marketing plan development," in *2017 International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT)*, pp. 238–242, 2017.

[24] I. Hamieh, R. Myers, and T. Rahman, "Construction of autonomous driving maps employing lidar odometry," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pp. 1–4, 2019.

[25] J. Liu, Q. Sun, Z. Fan, and Y. Jia, "Tof lidar development in autonomous vehicle," in *2018 IEEE 3rd Optoelectronics Global Conference (OGC)*, pp. 185–190, 2018.

[26] J. Kocić, N. Jovičić, and V. Drndarević, "Sensors and sensor fusion in autonomous vehicles," in *2018 26th Telecommunications Forum (TELFOR)*, pp. 420–425, 2018.

[27] M. Li and D. Yin, "A fast segmentation method of sparse point clouds," in *2017 29th Chinese Control And Decision Conference (CCDC)*, pp. 3561–3565, 2017.

[28] H. Zhang, B. Zhuang, and Y. Liu, "Object classification based on 3d point clouds covariance descriptor," in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 2, pp. 234–237, 2017.

[29] R. Huang, Y. Xu, and U. Stilla, "Extraction of multi-scale geometric features for point cloud classification," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2499–2502, 2019.

[30] Y. Lee and S. Seo, "Real-time object tracking in sparse point clouds based on 3d interpolation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4804–4811, 2018.

[31] N. Akai, Y. Morales, E. Takeuchi, Y. Yoshihara, and Y. Ninomiya, "Robust localization using 3d ndt scan matching with experimentally determined uncertainty and road marker matching," pp. 1356–1363, 06 2017.

[32] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," 2020.

[33] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low drift, robust, and fast," in *IEEE International Conference on Robotics and Automation(ICRA)*, (Seattle, WA), May 2015.

[34] S. G. Tzafestas, "12 - mobile robot localization and mapping," in *Introduction to Mobile Robot Control* (S. G. Tzafestas, ed.), pp. 479 – 531, Oxford: Elsevier, 2014.

[35] D. Galar and U. Kumar, "Chapter 1 - sensors and data acquisition," in *eMaintenance* (D. Galar and U. Kumar, eds.), pp. 1 – 72, Academic Press, 2017.

[36] J. F. Chow, B. B. Kocer, J. Henawy, G. Seet, Z. Li, W. Y. Yau, and M. Pratama, "Toward underground localization: Lidar inertial odometry enabled aerial robot navigation," in *Workshop on Challenges in Vision-based Drones Navigation (IROS)*, 2019.

[37] A. Noda, Y. Yamakawa, M. Ishikawa, and M. Shimojo, "Integration of high-speed visual and tactile sensors with synchronization in a sensor network system," in *2016 IEEE SENSORS*, pp. 1–3, 2016.

[38] C. Le Gentil, T. Vidal-Calleja, and S. Huang, "3d lidar-imu calibration based on upsampled preintegrated measurements for motion distortion correction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2149–2155, 2018.

[39] H. Nouira, J.-E. Deschaud, and F. Goulette, "Point cloud refinement with a target-free intrinsic calibration of a mobile multi-beam lidar system," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B3, pp. 359–366, 06 2016.

[40] J.-L. Blanco, "A tutorial on SE(3) transformation parameterizations and on-manifold optimization," Tech. Rep. 012010, University of Malaga, 2010.

[41] P. van den Bosch and A. van der Klauw, *Modeling, Identification and Simulation of Dynamical Systems*. Taylor & Francis, 1994.

[42] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 829–846, 2018.

[43] H. Wang, J. Wang, L. Qu, and Z. Liu, "Simultaneous localization and mapping based on multilevel-ekf," in *2011 IEEE International Conference on Mechatronics and Automation*, pp. 2254–2258, 2011.

[44] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for imu calibration without external equipments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3042–3049, 2014.

[45] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4304–4311, 2016.

[46] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," *2019 International Conference on Robotics and Automation (ICRA)*, May 2019.

[47] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, "Sensor fusion for joint 3d object detection and semantic segmentation," *CVPR*, 2019.

[48] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard, "Monocular camera localization in 3d lidar maps," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1926–1931, 2016.

[49] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences," in *Proceedings of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS*, October 2018.

[50] "Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving, howpublished = https://github.com/apolloauto/apollo, note = Accessed: 2019-02-11."

[51] H. Lategahn and C. Stiller, "Vision-only localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1246–1257, 2014.

[52] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Proceedings of Robotics: Science and Systems Conference*, July 2014.

[53] Y. Kim, J. Jeong, and A. Kim, "Stereo camera localization in 3d lidar maps," in *2018*

IEEE/RSJ International Conference on Intelligent Robots and Systems (*IROS*), pp. 1–9, 2018.

[54] J. Deray, J. Solà, and J. Andrade-Cetto, "Joint on-manifold self-calibration of odometry model and sensor extrinsics using pre-integration," in *2019 European Conference on Mobile Robots* (*ECMR*), pp. 1–6, 2019.

[55] A. S. Corporation, *Applied Optimal Estimation*. The MIT Press, 1974.

[56] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," *SPIE Proceedings*, p. 12, July 1997.

[57] V. Madyastha, V. Ravindra, S. Mallikarjunan, and A. Goyal, "Extended kalman filter vs. error state kalman filter for aircraft attitude estimation," *AIAA Guidance, Navigation, and Control Conference 2011*, 08 2011.

[58] A. Santamaria-Navarro, J. Solà, and J. Andrade-Cetto, "High-frequency mav state estimation using low-cost inertial and optical flow measurement units," *IEEE*, p. 8, December 2015.

[59] V. Sushrutha Raghavan, D. Kanoulas, C. Zhou, D. G. Caldwell, and N. G. Tsagarakis, "A study on low-drift state estimation for humanoid locomotion, using lidar and kinematic-inertial data fusion," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots* (*Humanoids*), pp. 1–8, 2018.

[60] J. Solà, "Quaternion kinematics for the error-state kalman filter," 2017.

[61] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (*Adaptive Computation and Machine Learning*). The MIT Press, 2005.

[62] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[63] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, p. 408–423, Feb 2015.

[64] J. Yoo, H. J. Kim, and K. H. Johansson, "Path planning for remotely controlled uavs using gaussian process filter," in *2017 17th International Conference on Control, Automation and Systems* (*ICCAS*), pp. 477–482, 2017.

[65] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, p. 1319–1340, Sep 2018.

[66] J. C. Platt, C. J. C. Burges, S. Swenson, C. Weare, and A. Zheng, "Learning a gaussian process prior for automatically generating music playlists," in *Advances in Neural Information Processing Systems 14* (T. G. Dietterich, S. Becker, and Z. Ghahramani, eds.), pp. 1425–1432, MIT Press, 2002.

[67] J. Gonzalvez, E. Lezmi, T. Roncalli, and J. Xu, "Financial applications of gaussian pro-

cesses and bayesian optimization," 2019.

[68] E. Abbasnejad, *Scalable Loss-calibrated Bayesian Decision Theory and Preference Learning*. PhD thesis, School of Computer Science, The Australian National University, 2017.

[69] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, p. 1–139, August 2017.

[70] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. Kschischang, "The factor graph approach to model-based signal processing," *Proceedings of the IEEE*, vol. 95, pp. 1295 – 1322, 07 2007.

[71] H. Ye, Y. Chen, and M. L. Liu, "Supplementary material to : Tightly coupled 3 d lidar inertial odometry and mapping," 2018.

[72] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, p. 381–395, June 1981.

[73] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," in *Computer Vision – ECCV 2008* (D. Forsyth, P. Torr, and A. Zisserman, eds.), (Berlin, Heidelberg), pp. 500–513, Springer Berlin Heidelberg, 2008.

[74] "Robust linear model estimation using ransac."

[75] R. Zeineldin and N. El-Fishawy, "A survey of ransac enhancements for plane detection in 3d point clouds," vol. 26, pp. 519–537, 07 2017.

[76] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3." http://eigen.tuxfamily.org, 2010.

[77] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

[78] D. Koguciuk, "Parallel ransac for point cloud registration," *Foundations of Computing and Decision Sciences*, vol. 42, 09 2017.

[79] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system."

[80] nw2190, "Cppgp - c++ gaussian process library," 01 2019. Implementation of Numerical Gaussian Processes in C++.

[81] S. Agarwal, K. Mierle, and Others, "Ceres solver." http://ceres-solver.org.

[82] S. Medina, P. Lancheros, L. Sanabria, N. Velasco, and L. Solaque, "Localization and mapping approximation for autonomous ground platforms, implementing slam algorithms," in *2014 III International Congress of Engineering Mechatronics and Automation (CIIMA)*, pp. 1–5, 2014.

[83] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[84] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[85] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[86] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *IROS*, vol. 3, pp. 2743 – 2748 vol.3, 11 2003.

[87] M. Magnusson, *The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection*. PhD thesis, Örebro University, 12 2009.

[88] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," 2018.

[89] R. Howe, "Very basic lie theory," *The American Mathematical Monthly*, vol. 90, no. 9, pp. 600–623, 1983.

[90] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

[91] J. G. Mangelson, M. Ghaffari, R. Vasudevan, and R. M. Eustice, "Characterizing the uncertainty of jointly distributed poses in the lie algebra," 2019.

[92] E. Eade, "Lie groups for 2d and 3d transformations," May 2017.

[93] J. G. Mangelson, M. Ghaffari, R. Vasudevan, and R. M. Eustice, "Characterizing the uncertainty of jointly distributed poses in the lie algebra," *IEEE Transactions on Robotics (TRO)*, 2019.

[94] Y. Pan, "Target-less registration of point clouds: A review," 2019.

[95] B. Ceulemans, S. Lu, P. Schelkens, and A. Munteanu, "Globally optimized multiview video color correction using dense spatio-temporal matching," in *2015 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–4, 2015.

[96] L. Fei, L. Yan, C. Chen, Z. Ye, and J. Zhou, "Ossim: An object-based multiview stereo algorithm using ssim index matching cost," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 12, pp. 6937–6949, 2017.

[97] U. Katsura, K. Matsumoto, A. Kawamura, T. Ishigami, T. Okada, and R. Kurazume, "Spatial change detection using normal distributions transform," *ROBOMECH Journal*, vol. 6, 12 2019.

[98] L. Markley, Y. Cheng, J. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 1193–1196, 07 2007.

[99] P. Cirujeda, Y. Dicente Cid, H. Müller, D. Rubin, T. Aguilera, B. Loo, M. Diehn, X. Binefa, and A. Depeursinge, "A 3-d riesz-covariance texture model for prediction of nodule recurrence in lung ct," *IEEE Transactions on Medical Imaging*, vol. 35, 07 2016.

[100] W. Zhen, S. Zeng, and S. Soberer, "Robust localization and localizability estimation with a rotating laser scanner," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6240–6245, 2017.

[101] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4531–4537, 2011.

[102] koide3, "Multi-threaded and sse friendly ndt algorithm," October 2017.

[103] "Bayesian filtering library with real-time support ("fl")."

[104] "Digitising construction."

[105] "Boldly go where no robot has gone before."

[106] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[107] V. L. Inc., *VLP-16 User Manual*, rev. d ed., 2018.

[108] "3dm-gx5-45 gnss/ins: High performance gnss navigation sensor, general package."

[109] "Velodyne hdl-64e."

[110] "Oxts rt3000 v3: Our highest performance ins for adas and autonomous vehicle testing."

[111] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, 09 2013.

[112] thomas789, "kitti2bag: Convert kitti dataset to ros bag file the easy way!."

[113] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, 2004.

[114] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *Proc. of the IEEE International Conference on Robotics and Automation*, (Paris, France), 2020.

[115] laboshinl, 2015.

[116] H. Ye, Y. Chen, , and M. Liu, "A tightly coupled 3d lidar and inertial odometry and mapping approach," 2019.

[117] H. Nguyen, F. Mascarich, T. Dang, and K. Alexis, "Autonomous aerial robotic surveying and mapping with application to construction operations," 2020.

[118] M. Beul, D. Droeschel, M. Nieuwenhuisen, J. Quenzel, S. Houben, and S. Behnke, "Fast autonomous flight in warehouses for inventory applications," *IEEE Robotics and Automation Letters*, vol. 3, p. 3121–3128, Oct 2018.

[119] D.-C. Hoang, T. Stoyanov, and A. J. Lilienthal, "Object-rpe: Dense 3d reconstruction and pose estimation with convolutional neural networks for warehouse robots," 2019.

[120] F. Kraemer, A. Schaefer, A. Eitel, J. Vertens, and W. Burgard, "From plants to landmarks: Time-invariant plant localization that uses deep pose regression in agricultural fields," 2017.

[121] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, p. 58443–58469, 2020.

[122] A. Bhutani and P. Wadhwani, "Lidar market size by product (airborne [topographic, bathymetric], uav, terrestrial [mobile, static]), by type (mechanical, solid-state), by application (corridor mapping [roadways, railways, bridges & tunnels], engineering, environment, driverless cars, exploration & meteorology, cartography), industry analysis report, regional outlook, application development, competitive landscape & forecast, 2019 - 2025," tech. rep., Global Market Insights, 04 2019.

[123] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," 2020.

[124] P. G. Esteban, D. H. García, H. R. Lee, P. Chevalier, P. Baxter, C. L. Bethel, J. Shukla, J. Oliver, D. Puig, J. R. Wilson, L. Tickle-Degnen, M. Bartlett, T. Belpaeme, S. Thill, K. Baraka, F. S. Melo, M. Veloso, D. Becerra, M. Matarić, E. Fosch-Villaronga, J. Albo-Canals, G. Beraldo, E. Menegatti, V. D. Tommasi, R. Mancin, F. Benini, Z. Henkel, K. Baugus, D. C. May, L. Dupuy, W. A. Rogers, R. F. Polak, S. Levy-Tzedek, D. Cruz-Sandoval, J. Favela, M. J. Johnson, M. Mohan, and R. Mendonca, "Proceedings of the workshop on social robots in therapy: Focusing on autonomy and ethical challenges," 2018.

[125] R. Kawamoto, H. Mochizuki, Y. Moriguchi, T. Nakano, M. Motohashi, Y. Sakai, and A. Inaba, "Estimation of co2 emissions of internal combustion engine vehicle and battery electric vehicle using lca," *Sustainability*, vol. 11, p. 2690, 05 2019.

[126] J. Worland, "Self-driving cars could help save the environment—or ruin it. it depends on us," 09 2016.

[127] A. Wang, "Self-driving cars are the future, but how do they impact the environment?," 07 2019.

[128] N. Puri and Y. Turkan, "Bridge construction progress monitoring using lidar and 4d design models," *Automation in Construction*, vol. 109, p. 102961, 2020.

[129] J. Hecht, "Automotive lidar: Safety questions raised about 1550 nm lidar," 09 2019.

[130] M. Brossard, S. Bonnabel, and A. Barrau, "Unscented kalman filter on lie groups for visual inertial odometry," pp. 649–655, 10 2018.

[131] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant ex-

tended kalman filtering for robot state estimation," 2019.